Don't Panic

GUIA A LA GALAXIA DE APLICACIONES MÓVILES

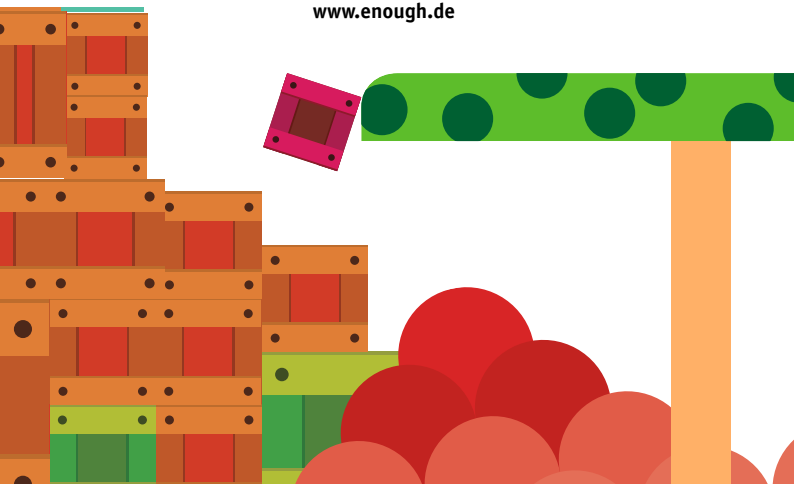


publicado por:

ENOUGH SOFTWARE

We create Apps.

Enough Software GmbH + Co. KG
Stavendamm 22
28195 Bremen
Alemania
www.enough.de





16a edición Febrero 2016

Esta guía se encuentra bajo la licencia Creative Commons
"Algunos derechos reservados."

Por favor, envía tus aportaciones, preguntas o esponsorización a:
mobiledevguide@enough.de
Síguenos en Twitter: *@MobileDevGuide*

Diseño:

Cornelius Kwietniak
Mladenka Vrdoljak

Editores:

Marco Tabor
Mladenka Vrdoljak

www.mobiledevelopersguide.com

Guia a la galaxia de aplicaciones móviles
Contento

1 Prólogo

4 La Galaxia Móvil: Introducción
por Robert Virkus & Marco Tabor

19 De la Idea al Concepto
por Sebastian Meyer

29 Experiencia de Usuario y Diseño de Interfaz
por Anna Alfut

44 Android
por Andre Schmidt & Vikram Kriplaney

64 iOS
por Alex Repty

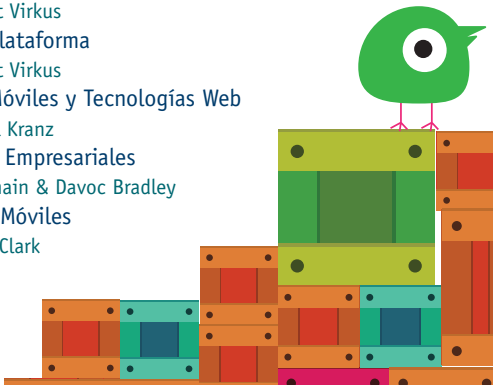
79 Windows
por Robert Virkus

90 Hacia Multiplataforma
por Robert Virkus

103 Sitios para Móviles y Tecnologías Web
por Daniel Kranz

121 Aplicaciones Empresariales
por Ian Thain & Davoc Bradley

134 Juegos para Móviles
por Oscar Clark





162 Desarrollo Móvil e Internet de las Cosas
por Alex Jonsson

173 Aplicaciones para Wearables
por Robert Virkus

182 Seguridad
por Dean Churchill

195 Accesibilidad
por Sally Cain

218 Testeo
por Julian Harty & Marc van't Veer

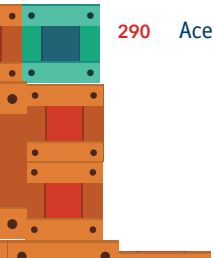
245 Analíticas Móviles
por Julian Harty

259 Recopilación y Comprensión del Feedback de Usuario
por Julian Harty

270 Monetización
por Michel Shuqair

289 Epílogo

290 Acerca de este Libro





Prólogo

Otro año, otra gran ronda de cambios: Mozilla terminó Firefox OS y Jolla pasó por una crisis financiera que casi puso fin al proyecto. BlackBerry lanzó su primer dispositivo Android (pero prometió continuar su apoyo a BlackBerry 10). Google es ahora una parte de su propio grupo de empresas "Alphabet". Apple lanzó Swift como código abierto y Microsoft lanzó su "sistema operativo como servicio" Windows 10. Así que el universo móvil sigue evolucionando y comienza a fusionarse con el sector de la tecnología ponible (o wearables).

Debido a estos cambios hemos decidido decir adiós en este libro a los capítulos dedicados a Firefox OS, Java ME, BlackBerry 10 y Tizen y concentrarnos aún más en temas estratégicos y conceptuales. Uno de los principales avances de la industria del móvil en los últimos tiempos es que el usuario está cada vez más en el centro de atención. Los desarrolladores y proveedores de aplicaciones han entendido que el feedback de los usuarios es un factor clave de éxito. Así que hemos ampliado el contenido del libro relacionado con el usuario aún más con esta edición: el concepto y los capítulos de diseño van perfilando cómo involucrar temprano a los usuarios y un capítulo totalmente nuevo acerca del feedback del usuario explica cómo tratar con tus clientes finales una vez has publicado.



Nos gustaría dar las gracias a nuestros dedicados autores, ¡sois geniales!

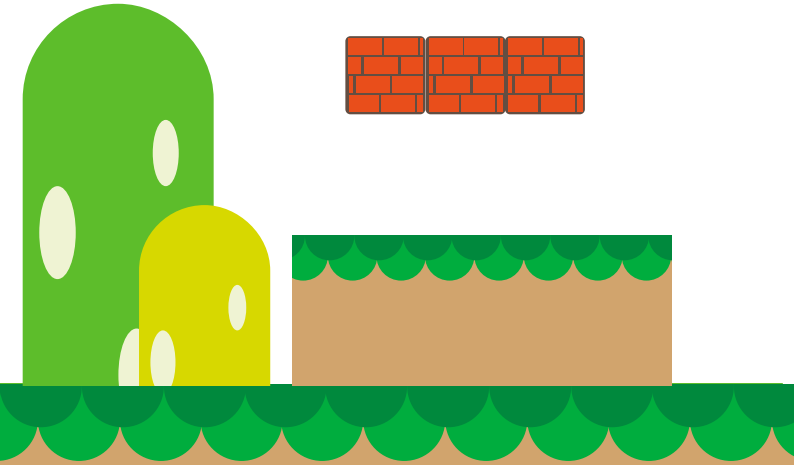
Otro gran agradecimiento va para Microsoft, nuestro patrocinador para la versión impresa (visita por favor sus ofertas para desarrolladores en *dev.windows.com* y *azure.com*).

Y ahora sigue adelante: ¡sumérgete en el universo móvil y déjanos ayudarte a encontrar tu sitio en él!

Robert + Marco / Enough Software

Bremen, Febrero 2016

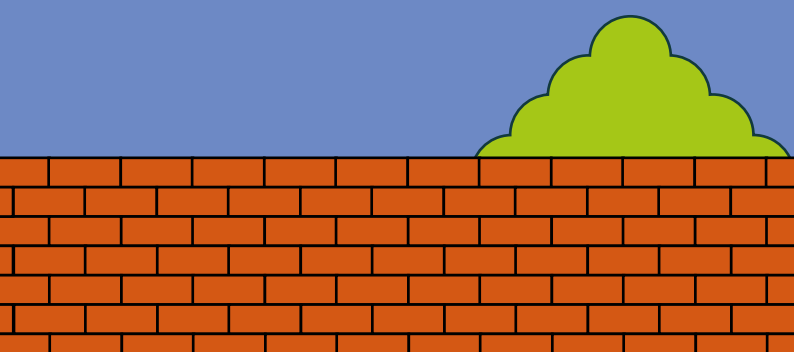
PD: Por favor, síguenos en Twitter *@MobileDevGuide* y visita mobiledevelopersguide.com para obtener la edición electrónica de este libro, disponible en varios idiomas.



MOBILE DEVELOPER'S GUIDE TO THE GALAXY

1 PLAYER GAME

2 PLAYER GAME



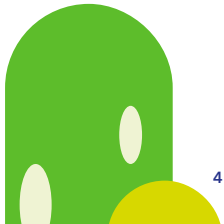
La Galaxia Móvil: Introducción

Bienvenido al mundo del desarrollo móvil, un lugar donde antiguos gigantes se tambalean y nacen nuevas estrellas, aparentemente de forma habitual.

Este libro se centra en el desarrollo de aplicaciones móviles, abarcando fases como: planificación y especificación, prototipado y diseño, implementación, testeo interno y despliegue, publicación en una tienda de aplicaciones, descubrimiento por parte de los usuarios, instalación, uso y feedback. En última instancia, queremos que nuestros usuarios disfruten utilizando nuestras aplicaciones y que nos den valoraciones positivas para alentar a otros a hacer lo mismo.

Sigue leyendo para aprender cómo desarrollar aplicaciones para las principales plataformas. Si esta es la primera vez que has considerado involucrarte en este tema, te recomendamos no demorarlo más; la tecnología móvil se ha convertido ya en la forma predominante de la informática en muchas áreas. El tiempo que pasamos utilizando aplicaciones móviles incluso ha superado ya al del televisor[1], al menos en los Estados Unidos.

Aunque el desarrollo de aplicaciones móviles tiene algunos puntos en común con el de otro tipo de software, también posee características específicas. Vamos a tratar algunas de ellas a continuación.



1 dminc.com/blog/mobile-app-usage-surpasses-tv

Topología: Factores de Formato y Patrones de Uso

Tradicionalmente, los desarrolladores de aplicaciones únicamente nos habíamos centrado en los teléfonos móviles; posteriormente las tablets aparecieron en escena como plataformas objetivo; y hoy en día nuestras aplicaciones pueden cubrir un amplio espectro de dispositivos: smartwatch, gafas inteligentes, móvil, tablet, PC, TV y automoción. Y otros factores de forma se incorporarán en el futuro. Cada factor de forma plantea sus propios retos de usabilidad; por ejemplo, un tablet exige una navegación diferente a un teléfono y la entrada de datos en un televisor puede resultar muy incómoda.

Naturalmente, los patrones de uso de Android difieren de los de iOS, que a su vez son diferentes de aquellos para Windows, etcétera. Por tanto, deberías evitar ofrecer la misma experiencia de usuario en todos los factores de forma, incluso si tu plataforma objetivo se limita a los smartphones, de lo contrario corres el riesgo de ofrecer un servicio mediocre a algunos segmentos de tus base de usuarios.

Formación Estelar: Creando un Servicio Móvil



Hay varias maneras de producir un servicio móvil:

- Aplicación (app)
- Sitio web
- SMS, USSD[2] y STK[3]

2 en.wikipedia.org/wiki/USSD

3 en.wikipedia.org/wiki/SIM_Application_Toolkit

Aplicaciones

Las aplicaciones se ejecutan directamente en el dispositivo. Se pueden desarrollar como nativas, web o híbridas.

Aplicaciones Nativas

Una aplicación nativa está programada en un lenguaje específico con APIs propias de la plataforma. Se suele comprar, descargar y actualizar a través de la tienda de aplicaciones específica de la plataforma. Las aplicaciones nativas suelen ofrecer mejor rendimiento, integración más completa y la mejor experiencia de usuario en comparación con otras opciones; sin embargo, el desarrollo nativo suele ser también la opción de desarrollo más compleja.

Aplicaciones Web

Una aplicación web (web-app/web-based) se basa en HTML5, JavaScript y CSS, y no depende de ninguna tienda de aplicaciones. Es un sitio almacenado localmente en el móvil que trata de emular el aspecto y comportamiento (look-and-feel) de una aplicación.

Un ejemplo famoso de aplicación web es la del Financial Times[4], que abandonó la tienda de aplicaciones con el fin de evitar compartir los ingresos de los suscriptores; en cambio, la aplicación web de Facebook para iOS fue puesta al día como aplicación nativa para mejorar dramáticamente su rendimiento y usabilidad. Hay frameworks para aplicaciones web que permiten construir un contenedor nativo para tales aplicaciones de manera que puedan ser publicadas en las tiendas de aplicaciones, por ejemplo Phonegap[5].

4 apps.ft.com/home/web-app

5 www.phonegap.com

Aplicaciones Híbridas

Para muchos desarrolladores de aplicaciones móviles, se ha vuelto bastante común adoptar un enfoque híbrido para el desarrollo de aplicaciones: una aplicación puede utilizar código nativo para incrementar su rendimiento e integrarse con la plataforma al mismo tiempo que utiliza una webview con contenido basado en HTML5 para otras partes de dicha aplicación. Una aplicación híbrida hace uso tanto de las tecnologías nativas como de las web. Partes de ella se comportarán como una aplicación nativa, mientras que otras se ejecutan sobre tecnologías web y pueden utilizar la conexión a internet para ofrecer contenidos actualizados y funcionalidades sin necesidad de publicar actualizaciones en las tiendas de aplicaciones. Aunque esto podría ser visto como un inconveniente, el uso de las tecnologías web permite a los desarrolladores mostrar contenido actualizado sin tener que volver a subir la aplicación a las tiendas de aplicaciones. El desafío clave es combinar las capacidades únicas de tecnologías nativas y web para crear una aplicación verdaderamente fácil de usar y atractiva.

Sitios Web

Un sitio web se ejecuta en tu servidor, pero puedes acceder a varias funciones del dispositivo con JavaScript, por ejemplo para almacenar datos de forma local o solicitar la ubicación actual del dispositivo. A diferencia de las aplicaciones, los sitios web para móviles son inherentemente multiplataforma. Históricamente, los sitios web se han centrado en ser compatibles con navegadores basados en WebKit como Safari. Hoy en día, tanto WebKit como Chromium son los motores de renderizado web dominantes, con Trident de Internet Explorer y algunos otros como actores secundarios. En aras de una web abierta, intenta usar estándares HTML5 tanto como sea posible y evita renderizar código específico de un único motor.

SMS, USSD y STK

Los servicios sencillos pueden ser realizados con SMS, USSD o STK. Todo el mundo sabe cómo funciona el sistema de mensajería de texto SMS (Short Message Service), y todos los teléfonos son compatibles con él, pero necesitas convencer a los usuarios de que recuerden comandos de texto para servicios más complejos. En el Reino Unido, se puede pagar en los parkings de muchas ciudades vía SMS.

El USSD (Servicio Suplementario de Datos no Estructurados, del inglés Unstructured Supplementary Service Data) es un protocolo GSM utilizado para enviar sencillos menús basados en texto, cuyas capacidades dependen de la compañía operadora y el dispositivo. En Sri Lanka, los turistas pueden recibir una tarjeta SIM gratuita que se registra utilizando menús USSD.

El STK (SIM Application Toolkit) permite implementar aplicaciones interactivas de bajo nivel directamente en la tarjeta SIM de un teléfono. El STK puede parecer irrelevante cuando se está dando tanta relevancia a las aplicaciones para smartphones, sin embargo hay casos como el de M-Pesa, una aplicación STK que está transformando las transacciones financieras en Kenia y otros países.[6]

El Universo de los Sistemas Operativos Móviles

El espacio móvil es mucho más diverso que otras áreas TIC. Cuando estás desarrollando software para ordenadores personales, tienes básicamente tres sistemas operativos para elegir, pero en el caso de los móviles hay muchos más. Este libro te ofrece una introducción a aquellos sistemas operativos móviles más relevantes actualmente, pero hay que ser conscientes de

6 mpesa.in

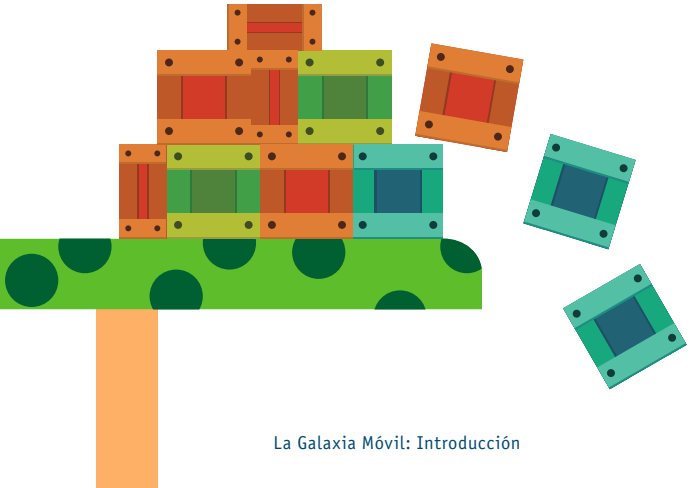
que el entorno móvil cambia continuamente a una velocidad que rara vez se observa en otros negocios. Hemos visto cómo muchas tecnologías prometedoras aparecen y desaparecen rápidamente, sin importar lo grandes que eran las compañías que las respaldaban o la relevancia que tuvieran históricamente en el mercado.

Así que sigue leyendo, aprende cómo es el mercado actual y prepárate para seguir sus cambios (o asegúrate de tener la última edición de esta guía a mano).

Cuásars: Android e iOS

Cuando la gente habla sobre aplicaciones móviles, en su mayoría solo se refieren a Android e iOS. ¿Por qué? En términos de cuota de mercado, estas dos plataformas combinadas dominan el mercado de smartphones alcanzando fácilmente el 90% de cuota en los principales mercados[7] (ver la tabla a continuación para los números globales). El estudio Developer

7 www.theverge.com/2015/8/20/9181269/gartner-q2-2015-smartphone-sales



Economics del tercer trimestre del 2015[8] también muestra que iOS y Android están a la cabeza en términos de atención por parte de los desarrolladores, es decir, el porcentaje de desarrolladores utilizando una plataforma, independientemente de la plataforma que ellos consideran su 'primaria'. Android era líder, con un 71% de los desarrolladores trabajando en ella, seguido por iOS con un 51%.

Naturalmente, esto también significa que si vas a usar Android o iOS vas a tener muchos competidores.

Magnetoestrella: Windows

Aunque Windows Phone ha tenido algunos éxitos, su cuota de mercado en todo el mundo sigue siendo baja, demasiado baja para muchos desarrolladores de aplicaciones. Así que Microsoft cambió las reglas con Windows 10: ahora puedes desarrollar la misma aplicación para PC y móvil (y también para la Internet de las Cosas -IoT, del inglés Internet of Things-, HoloLens y Xbox). Hasta ahora parece que esta estrategia merece la pena ya que varios actores importantes se han unido al ecosistema de Windows últimamente.

Para saber más acerca de la plataforma y cómo empezar, consulta el capítulo correspondiente en este libro.

Nueva Estrella: Ubuntu

Los dispositivos móviles que funcionan con Ubuntu han estado entrando en el mercado en 2015. Con su concepto innovador y la naturaleza de código abierto ha generado cierto interés en las comunidades de desarrolladores, pero hasta el momento esto parece tener poca influencia en el volumen de ventas. Uno de los principales problemas percibidos es el problema de

8 DeveloperEconomics.com

rendimiento, incluso en la más alta gama de hardware como el Meizu MX4.

Tu principal punto de entrada es ubuntu.com/phone/developers. Puedes desarrollar tus aplicaciones ya sea como aplicaciones web o de forma nativa utilizando Qt/C++. Un concepto interesante son los ámbitos, que permiten a los usuarios ver información sin necesidad de iniciar tu aplicación de forma explícita. Se han anunciado frameworks que permitirían participar e integrarse en actividades básicas como la mensajería.

Nueva Estrella: Tizen

Tizen[9] ha disfrutado de un cierto éxito en el mercado de smartwatches (ver el capítulo dedicado a estos dispositivos), aunque se ha retrasado la producción de los teléfonos que lo soportarían prometidos por Samsung. El Z1, el primer teléfono Tizen, ha sido finalmente lanzado en India en enero del 2015. Aparentemente impulsado de manera suave pero firme por Samsung e Intel, Tizen aspira a soportar no solo teléfonos y smartwatches, sino también televisores, tablets, netbooks y sistemas de información y entretenimiento para vehículos.

Las aplicaciones Tizen típicas están basadas en la web, pero también se pueden crear aplicaciones nativas basadas en C. Comienza tu viaje en Tizen en *developer.tizen.org* y *developer.samsung.com/gear*.

Estrella Oscura: BlackBerry 10

A finales del 2015 BlackBerry presentó su primer teléfono Android, el BlackBerry Priv. Al mismo tiempo BlackBerry se comprometió a seguir apoyando BlackBerry 10[10]. Pero la

9 tizen.org

10 devblog.blackberry.com/2015/10/an-update-for-blackberry-10-developers

pregunta sigue siendo si BlackBerry puede seguir el ritmo de dar soporte a dos plataformas al mismo tiempo.

Puedes desarrollar aplicaciones para BlackBerry 10 con Qt/C++, tecnologías web o, gracias a su entorno de ejecución compatible con Android, incluso como aplicaciones de Android. Empieza tu viaje BlackBerry en *developer.blackberry.com*.

Supernova: Sailfish OS

¿Explotará? Jolla[11] (la compañía detrás de Sailfish OS[12]) entró en una reestructuración de la deuda[13] y despidió[14] a alrededor de la mitad de sus empleados. Si bien las perspectivas futuras pueden parecer algo oscuras, todavía no está todo perdido: otra ronda de financiación ha salvado el sistema operativo por ahora, pero tuvieron que renunciar a los planes para entrar en producción en masa con su propia tablet[15].

Empieza a desarrollar para Sailfish OS visitando sailfishos.org/develop. Una ruta de acceso fácil para Sailfish consiste en utilizar tu aplicación Android ya existente, pero también puedes crear aplicaciones nativas utilizando Qt/C++ o incluso Python.

Materia Oscura: Plataformas de Móviles de Gama Media

Si bien los smartphones acaparan todo el protagonismo en los medios, en algunas partes del mundo los móviles de gama media todavía tienen un papel bastante relevantes. Incluso a nivel global, el 22% de todos los teléfonos vendidos en el primer trimestre del 2015 han sido móviles de gama media[16],

11 jolla.com

12 sailfishos.org

13 reviewjolla.blogspot.de/2015/11/news-jolla-financial-trouble-debt.html

14 reviewjolla.blogspot.de/2015/11/news-jolla-financial-difficulties.html

15 blog.jolla.com/new-years-greetings-jolla

16 counterpointresearch.com/marketmonitor2015q1

con una base instalada muy superior a esa cifra. Aún así, su futuro no está del todo claro ya que Android está incrementando su penetración en el mercado de dispositivos de bajo coste.

Incluso los actores clave en el mercado de teléfonos de gama media se han dado cuenta: Nokia cerró su tienda de aplicaciones para móviles de gama media en 2015.

Aunque se pueden desarrollar aplicaciones nativas para teléfonos de gama media cuando se tiene estrecha relación con el fabricante, normalmente desarrollarás aplicaciones para esos teléfonos utilizando Java ME o BREW.

Enanas Blancas: Firefox OX, Symbian, bada y Otros Sistemas Muertos

Algunos sistemas operativos han ido consumiéndose lentamente (como Samsung bada), otros han terminado con una explosión (como WebOS), y algunos han sido reabastecidos gracias a nuevos desarrollos. El último de esta lista es Firefox OS[17]. Pese a que tuvimos incluso un capítulo dedicado a esta plataforma en nuestra última Guía del Desarrollador Móvil, Mozilla anunció en diciembre del 2015 que el desarrollo de Firefox OS en teléfonos móviles será descontinuado[18]. ¿Por qué algunos sobreviven mientras que otros fracasan? Al final todo se reduce al marketing, al reconocimiento por parte de la comunidad de desarrolladores, a las políticas corporativas y a una buena dosis de pura suerte. Cada vez es más difícil competir con el enorme peso que tienen los ecosistemas iOS y Android, veremos si esta tendencia continúa también en el sector de los wearables.

17 mozilla.org/firefox/os

18 techcrunch.com/2015/12/08/mozilla-will-stop-developing-and-selling-firefox-os-smartphones

Sistema Solar: Cuotas de Mercado de Sistemas Operativos para Smartphones

Si echas un vistazo a la cuota de mercado mundial de smartphones para el mismo trimestre durante años sucesivos, la imagen resulta muy clara:

Plataforma	Cuota T3 2014	Cuota T3 2013	Cuota T3 2012	Cuota T3 2011
Android	84,4%	81,2%	74,9%	57,4%
iOS (Apple)	11,7%	12,8%	14,4%	13,8%
Windows Phone	2,9%	3,6%	2,0%	1,2%
BlackBerry	0,5%	1,7%	4,1%	9,6%
Otros	0,6%	0,6%	4,5%	18,8%

(Fuente: *www.statista.com/statistics/266136/global-market-share-held-bysmartphone-operating-systems/*)

Es posible que, como la mayoría de los desarrolladores, consideres que invertir tiempo en todo lo que no sea iOS y Android es una pérdida de tiempo, pero te aseguramos que no es tan sencillo como parece. Aunque a nivel mundial las ventas de smartphones superan hoy a las de teléfonos de gama media[19], estos teléfonos de gama media todavía se venden más que los smartphones en algunas regiones e incluso crecen en popularidad, por ejemplo en Japón[20].

Recuerda también que estas son cifras globales; la participación de cada plataforma en los mercados regionales es muy variable. En un mundo donde el contenido geolocalizado

19 idc.com/getdoc.jsp?containerId=prUS24085413

20 reuters.com/article/japan-tech-mobilephone-idUSL4N0VM1HT20150216

está aumentando en importancia, es fundamental conocer los detalles y las características del mercado objetivo. Por ejemplo, China es el mayor mercado de smartphones en la actualidad, pero los dispositivos Android chinos están basados habitualmente en Android Open Source Platform (AOSP) y se comercializan sin acceso a Google Play y otros servicios Google para móviles. Al mismo tiempo, Apple está especialmente presente en los Estados Unidos: en octubre del 2015, el 43,3% de los usuarios de smartphones en Estados Unidos estaban utilizando un dispositivo iOS[21].

Para obtener información sobre la cuota de mercado en tu región objetivo, tienes recursos en línea como comscore[22], StatCounter[23], VisionMobile[24], Gartner[25], Statista[26] o Kantar Mobile World Panel[27].



21 Sin embargo, Android lidera también en los Estados Unidos con un 52,9% del mercado, según statista.com/statistics/266572/market-share-held-by-smartphone-platforms-in-the-united-states

22 www.comscore.com/Insights/Data-Mine

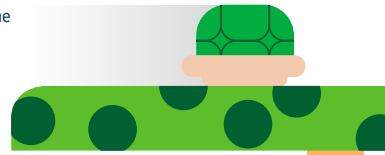
23 gs.statcounter.com

24 visionmobile.com

25 gartner.com

26 statista.com/markets/418

27 kantarworldpanel.com/global/smartphone-os-market-share





Sobre Tiempo y Espacio

Como desarrolladores, tendemos a mostrar mucha pasión por nuestros sistemas favoritos. Sin embargo no debemos olvidar que estas tecnologías son solo eso, tecnologías que son relevantes en un momento y lugar dado, pero nada más. Sí, algunas disputas en internet son divertidas pero, en retrospectiva, siempre son pueriles. ¡Que levanten la mano aquellos que participaron en el debate Atari versus Amiga en los viejos y buenos años 80! Probablemente no muchos de vosotros, pero seguramente entendéis a lo que nos referimos. Iniciativas como FairPhone[28], ShiftPhone[29] o GuardianProject[30] pueden ser más importantes que el sistema operativo o el proveedor que elijas en el futuro.

28 fairphone.com

29 shiftphones.com

30 guardianproject.info



Perdido en el Espacio

Si estás perdido en el vasto universo del desarrollo móvil, no te preocupes, mantén la calma y sigue leyendo. Explora tus opciones y evalúa el problema que deseas resolver, tu público objetivo y tu know-how. Pon un gran esfuerzo en diseñar la experiencia de tu servicio, concéntrate en el problema en cuestión, y mantenlo simple. Es mejor hacer una cosa bien que hacer "todo" solo regular. Invierte en el diseño y la usabilidad de la solución. Por último, pero no menos importante, encontrar el nicho de mercado adecuado es a menudo mejor que tratar de copiar algo que ya ha tenido éxito. ¡Esta guía te ayudará a tomar una decisión informada!



De la Idea al Concepto

Desarrollar soluciones digitales populares e innovadoras en esta industria significa beneficios importantes. Hay millones de apps en las tiendas de aplicaciones, muchas de ellas agrupadas en una masa de aplicaciones similares que no ofrecen nada especial. Sin embargo, algunas aplicaciones son excepcionales. Es posible que te hayas preguntado: ¿qué hace que estas aplicaciones tengan éxito, y cómo puedo lograr un éxito similar con mis productos? Este capítulo te ofrece herramientas y un marco para generar sistemáticamente ideas innovadoras para productos. Pero por favor, ten siempre en cuenta la posibilidad de que crear una aplicación pueda no tener sentido para tu proyecto y que podría ser mejor idea invertir tu dinero en otra parte. De lo contrario podrías terminar compartiendo la experiencia de Birdly, que resumen sus aprendizajes en un artículo titulado "Por qué no deberías preocuparte de crear una aplicación móvil"[1].

Los Ingredientes Clave del Éxito: Conveniencia, Viabilidad y Factibilidad

Comencemos debatiendo qué hace a un producto innovador y exitoso. Una característica importante de los productos innovadores es su combinación de tres aspectos principales: conveniencia humana, viabilidad financiera y viabilidad técnica.

Para comprender su conveniencia es necesario responder a las preguntas: ¿Qué desean las personas, y qué les hace felices

1 medium.com/inside-birdly/why-you-shouldn-t-bother-creating-a-mobile-app-328af62fe0e5#.9r624514i"

y les aporta valor? Un buen producto puede simplificar tareas o hacer más cómoda la vida de las personas. Por otro lado, una experiencia de usuario atractiva fomenta el placer de su uso, ya que las tareas se pueden realizar fácilmente, la estructura de la aplicación es muy clara y fácil de entender, y las actualizaciones frecuentes sobre el contenido o sitio ofrecido mantienen constante el interés en la misma. Las aplicaciones móviles pueden permitir a los usuarios hacer fácilmente cosas que no podían hacer antes; por ejemplo, comparar precios en una tienda, comprar y vender sobre la marcha, o controlar su salud y estado físico. Considera formas de desarrollar aplicaciones que aprovechen lo que ofrecen los dispositivos móviles.

La viabilidad financiera es crucial para el desarrollo de la mayoría de productos, incluyendo las aplicaciones móviles. La viabilidad ayuda a cubrir los costes en tiempo y energía del desarrollo y mantenimiento de tu aplicación. Para aquellos que esperan que la aplicación se autofinancie, el desafío es definir un modelo de negocio que permita crear ingresos a partir de una idea y mantener los costes en niveles aceptables para los clientes finales. En particular, en el mundo digital han surgido varios nuevos modelos de negocio en la última década. Por ejemplo, tanto la publicidad como el modelo freemium se han utilizado en las aplicaciones móviles, siendo las compras dentro de la aplicación especialmente populares en los videojuegos, aunque con riesgo de ser contraproducentes[2]. Herramientas como el Osterwalder Business Model Canvas[3] pueden ayudar a construir tu modelo de negocio de una manera estructurada. Lee el capítulo sobre monetización de esta guía para aprender más sobre qué opciones tienes de ganar dinero con el software móvil.

2 developereconomics.com/freemium-apps-killing-game-developers/

3 alexosterwalder.com

El tercer aspecto es la viabilidad técnica. La mayoría de los ingenieros de software y desarrolladores están involucrados en la evaluación de esta dimensión. A menudo, construir y combinar la tecnología adecuada para hacer un producto es todo un desafío. Casos de la vida real muestran que los productos innovadores no siempre necesitan tecnología de vanguardia para tener éxito y que una combinación inteligente de las tecnologías existentes puede generar productos innovadores.

Es importante tener en cuenta los tres aspectos (conveniencia, viabilidad y factibilidad) para desarrollar un concepto suficientemente detallado antes de invertir esfuerzos innecesarios implementando una solución. Obtén feedback de otras personas al principio, incluyendo los usuarios potenciales, para así ayudarte a refinar idea y concepto.

Define las Necesidades del Usuario

En teoría, es posible que un producto encaje inmediatamente con los deseos de tu grupo objetivo. No obstante, en la práctica esto es poco habitual. En la mayoría de casos el producto no agrada a muchos de los clientes previstos. Aunque las capacidades ofrecidas por el producto pueden ser muy innovadoras, frescas, y realmente útiles, los usuarios pueden considerar el producto no apto para su contexto o pueden necesitar varias funcionalidades adicionales para hacer un uso real del mismo. Para mejorar la satisfacción de los usuarios, las empresas de software tienden a hacer ajustes y tratar de responder a todos los deseos del grupo objetivo de una manera desorganizada. Consecuentemente, el producto se va volviendo complejo, no es muy usable, y pierde más y más usuarios, ya que las expectativas de los usuarios no se pueden cubrir con la aplicación existente.

Comenzar a desarrollar sin comprender realmente las necesidades del usuario es muy arriesgado porque los cambios

en una aplicación ya lanzada son caros. Además, se malgastan recursos y se crean funciones innecesarias.

Para evitar este tipo de fallos, es importante centrarse en el usuario y desarrollar la aplicación utilizando su feedback. Lo que se conoce como análisis de los requisitos de usuario describe los procesos cruciales de revelar las necesidades del usuario y determinar sus expectativas. Analizar el escenario del problema y comprender los requerimientos del usuario son partes integrales del diseño de soluciones digitales innovadoras.

El primer paso es saber quiénes son tus usuarios y definir el grupo o grupos de destino para tu aplicación. Tienes que comprender cuáles son los objetivos de los usuarios, qué tareas necesitan realizar y por qué tu aplicación es relevante para sus necesidades.

Con el fin de revelar los puntos problemáticos y deducir los requisitos reales, es necesario entender de qué manera realizan los usuarios las tareas actualmente, incluyendo sus soluciones alternativas. La mejor manera de obtener esta información es hablar directamente con los representantes de los grupos de usuarios y observarlos en su vida o trabajo diarios. Los estudios secundarios de mercado, por ejemplo informes o datos demográficos, pueden ampliar tu investigación, ¡pero por favor no confíes en ellos como fuente de información principal!

Además, durante el análisis de requisitos es importante tener en cuenta las diferencias entre deseos y necesidades. Según Merriam Webster, un deseo es "el anhelo de que algo suceda o sea hecho."[4]. Además, se trata de "un acto de pensar en algo que deseas, con la esperanza de que lo conseguirás o que va a suceder de alguna manera mágica". Habitualmente, este deseo es consciente. Los deseos se centran en objetos materiales concretos (por ejemplo los smartphones) o en

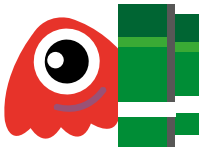
4 merriam-webster.com/dictionary/wish

habilidades (por ejemplo la creatividad). Es por esto que los usuarios pueden expresar sus deseos para un determinado producto, como colores o funciones especiales. Los deseos se pueden cambiar (por ejemplo vía publicidad o cuando productos mejores llegan a ser conocidos). Si un nuevo producto refleja solamente los deseos de usuarios, no significa necesariamente que les ayude con sus objetivos. Por ejemplo, algunos clientes quieren conseguir un producto únicamente porque tiene un diseño agradable.

En cambio, las necesidades se esconden tras los deseos. No se centran en objetos o habilidades, sino en factores emocionales, como el recibir reconocimiento, la confianza o una habilidad. Según Merriam Webster, una necesidad es "algo que una persona debe tener" y "algo que se necesita para vivir o tener éxito o ser feliz."[5]. Una necesidad es un deseo basado en la escasez. Si una persona experimenta la ausencia de algo, esto crea un deseo profundo en el subconsciente. Este deseo motiva los actos, lo que debería eliminar dicha ausencia. Diferentes carencias inducen diferentes niveles de acción. De acuerdo con la famosa pirámide de Maslow[6], la necesidad fisiológica de dormir o el hambre son, por ejemplo, mucho más fuertes que la necesidad de comunicación social. A diferencia de los deseos, las necesidades son no específicas y a menudo inconscientes.

Para iniciar el análisis de los requisitos, asegúrate de plantear necesidades y definir tu producto basándote en necesidades reales en lugar de deseos arbitrarios.





5 merriam-webster.com/dictionary/need

6 ver en.wikipedia.org/wiki/Maslow's_hierarchy_of_needs

Ideando

El primer resultado del análisis no es una solución, sino una declaración clara y bien fundamentada del problema. Este planteamiento debe ser una base para explorar el escenario de la solución en la fase de ideación del proyecto.

Durante la fase de ideación, es importante comenzar a pensar de manera muy divergente y llegar a una gran cantidad de ideas. Si creas una gran cantidad de ideas, es más probable que "pienses fuera de la caja" (del inglés thinking outside the box) y que surjan ideas realmente innovadoras para abordar el problema planteado.

Existen muchas técnicas para fomentar el pensamiento creativo. Algunas son bien conocidas, como el brainstorming (lluvia de ideas). Sin embargo, se han inventado muchas otras que se pueden utilizar en situaciones específicas, como el método 6-3-5[7], la confrontación visual, o el método Disney[8]. Lo que todas estas técnicas tienen en común es que fomentan tanto el pensamiento divergente como el convergente y también el "pensamiento fuera de la caja"".

Un consejo habitual consiste en celebrar sesiones de creatividad en un grupo con 5-8 participantes. Un grupo generalmente consigue mejores resultados que los de una persona individual. El clásico brainstorming es el método más popular, pero no es el que produce los mejores resultados en cuanto a la cantidad de ideas. Para mejorarlo, se recomienda la aplicación de técnicas de brainwriting, que permiten a cada participante recabar ideas individualmente antes de discutirlas en grupo.

7 una técnica de brainwriting para grupos, ver en.wikipedia.org/wiki/6-3-5_Brainwriting

8 desarrollado por Robert Dilts en 1994, ver en.wikipedia.org/wiki/Disney_method

Esto anima a cada miembro del equipo a participar activamente en la lluvia de ideas y reduce el riesgo de que estas se evalúen demasiado pronto y que las personas menos dominantes no sean escuchadas.

Independientemente de la técnica que se utilice, es importante tener en cuenta las siguientes reglas para el equipo durante la fase de ideación del proyecto:

- Aplazar el juicio
- Fomentar ideas extrañas
- Mantenerse centrado
- Priorizar la cantidad
- Ser visual
- Construir sobre las ideas de otros

Una vez recogidas las diversas ideas, deberán revisarse, retocarse o combinarse por el equipo, y las que resulten más prometedoras serán las seleccionados.

Probando las Ideas

Tras seleccionar una idea, es crucial descubrir si proporciona un valor real para el grupo destinatario y si realmente satisface las necesidades de los usuarios. Los comentarios de los usuarios potenciales son clave para evaluar el valor de tu idea. Al principio puedes simplemente hablar de tu idea y el concepto general de la aplicación. Sin embargo, puede ser complicado describir tus pensamientos de una manera en la que el público realmente los entienda de la misma manera que tú. El lenguaje natural se interpreta a menudo de manera diferente y crea diferentes modelos mentales en cada persona. Por tanto, la comprensión puede ser diversa y la comunicación de las ideas un reto.

Con el fin de superar los retos de la comunicación, debes expresar tus ideas de maneras visuales y tangibles, por ejemplo con prototipos. La idea de construir un prototipo puede sonarte como algo que requiere mucho esfuerzo. De hecho, tradicionalmente se entiende un prototipo como una versión previa de un producto final. Pero existe un enfoque más moderno del concepto que resulta más versátil al incluir borradores (por ejemplo, un dibujo en papel). En las primeras fases de desarrollo de una aplicación innovadora, un prototipo debe ser una herramienta que se utilice para discutir ideas y conceptos. Un prototipo puede ser, por ejemplo, un dibujo, guión gráfico, o la visualización física de un concepto. Debe ser fácil y rápido de construir y representar las partes esenciales de tu idea.

El principio más importante para trabajar con prototipos es "empezar con algo pequeño, fallar pronto y aprender rápido". Esto significa que puedes crear un prototipo rápido con costes muy bajos para obtener feedback sobre tu idea y aprender de este feedback en ciclos de iteración muy cortos. En esas iteraciones puedes probar y mejorar el prototipo rápidamente y a bajo coste. De esta manera, es posible validar nuevas ideas con riesgos bajos y sin costes adicionales para la ejecución del proyecto.

Cuando se trata de ingeniería de software, hay tres herramientas relacionadas con el término prototipo: wireframe, maqueta (en inglés, mockup) y prueba de concepto. Los wireframes y las maquetas se utilizan generalmente en las primeras fases para validar un concepto, arquitectura de la información o interacción básica. Implementar una prueba de concepto se utiliza para validar la viabilidad técnica. Durante la implementación de una prueba de concepto, es fundamental centrarse en los riesgos del proyecto e identificar soluciones y limitaciones técnicas. Lee el siguiente capítulo para obtener más información sobre cómo crear prototipos de aplicaciones.



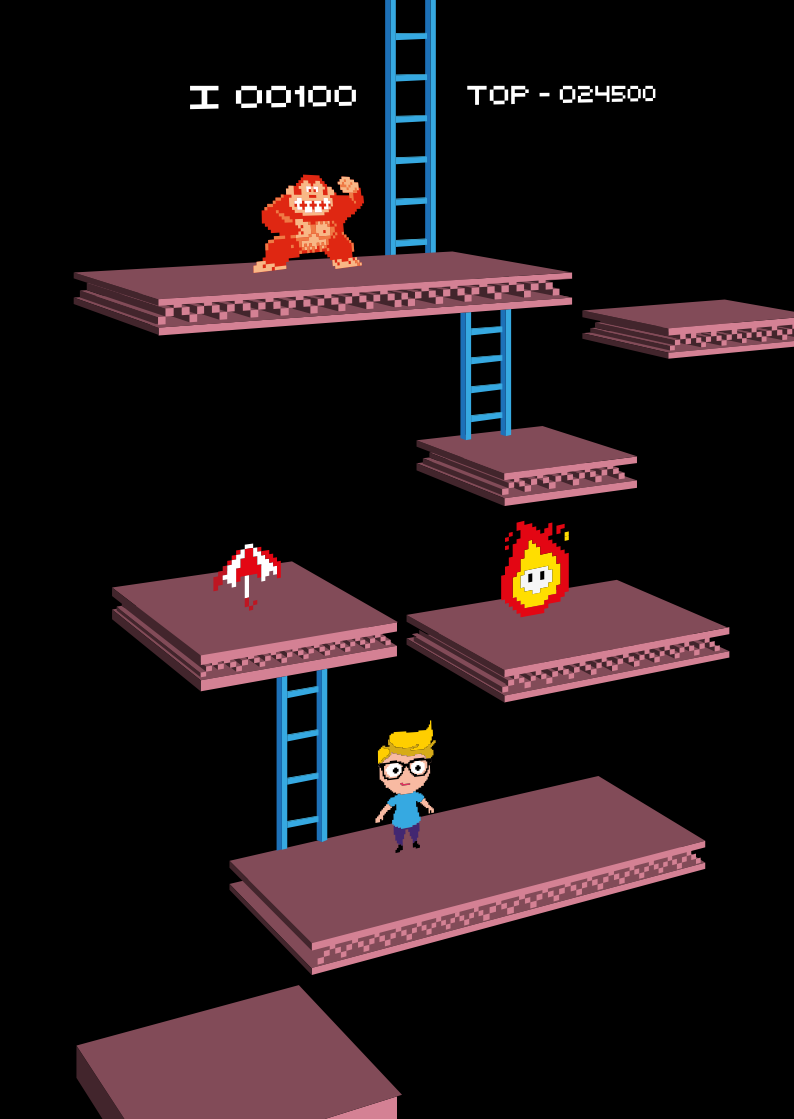
Aprender Más

Aquí puedes aprender más acerca de las metodologías y técnicas perfiladas anteriormente:

- *businessmodelgeneration.com*, el sitio web del famoso libro sobre cómo generar tu propio modelo de negocio. También ofrecen una herramienta online, una vista previa gratis del libro[9] y mucho más.
- *mycoted.com/Brainwriting*, una buena introducción a métodos de brainwriting. Forma parte de una wiki sobre técnicas creativas mantenida por la empresa inglesa mycoted.
- *uxbooth.com/articles/complete-beginners-guide-to-design-research*, la "Complete Beginner's Guide to Design Research" de Andrew Maier. El título es un tanto exagerado, pero el libro es una buena introducción con enlaces útiles a otros recursos.
- *theleanstartup.com*, el sitio web del libro "Lean Startup" de Eric Reis. Ayuda a los lectores a descubrir maneras de crear productos más apetecibles con menor coste.

9 businessmodelgeneration.com/book





BY Anna Alfut

Experiencia de Usuario y Diseño de Interfaz

A estas alturas probablemente tienes ya un concepto para tu aplicación. En el capítulo anterior has aprendido a generar y validar múltiples ideas. Esta sección trata sobre los métodos que son útiles para transformar tus conceptos en productos utilizables a través del diseño iterativo. Se centra en la descripción de una serie de técnicas para dar forma a tu idea. Puedes utilizarlas para esbozar las primeras ideas en un nivel alto, así como para definir las directrices detalladas para la aplicación final.

La experiencia de usuario (UX) trata de cómo tu propuesta es percibida generalmente por los usuarios finales. ¿Fue un camino fácil y rápido? ¿Tenían suficiente información en cada paso? Etcétera. Es un término amplio que se puede aplicar al diseño de servicios y productos de software. El diseño de la interfaz de usuario (UI) es, en cambio, un término que es específico para productos de software. Tanto si se trata de una página web como de una aplicación independiente, la interfaz de usuario es donde los usuarios van a experimentar lo que les ofreces a través de interacciones, diseño visual, flujo, mensajes, etc.

Diseñar una aplicación desde cero requiere un enfoque diferente al de mejorar un producto existente. En ambos casos tendrás que alternar a menudo entre una visión más general y una atención al detalle, para asegurarte de que sigues mejorando la calidad mientras te mantienes en el camino adecuado para completar tu idea.

Debes utilizar los métodos descritos en este capítulo como una caja de herramientas más que como una receta, y

aplicarlos en el orden que funcione mejor para el proyecto específico en el que estés trabajando. Si te encuentras en una fase muy temprana y solo necesitas alguna manera de delinear un concepto, puedes utilizar bosquejos o dibujar un mapa del trayecto de usuario. Cuando alcances otra iteración del diseño y necesites perfeccionar la interfaz de usuario, puedes utilizar los mismos métodos simplemente ajustando el grado de atención al detalle.

Las fases del diseño (pensamiento conceptual, definición de la interfaz y verificación de tus conceptos con usuarios) crean un bucle que se puede repetir tantas veces como sea necesario (o tantas como tiempo se disponga).

Experiencia de Usuario (UX)

Para dar a los usuarios de tu aplicación una experiencia que aprecien tienes que ser muy claro acerca de los objetivos de tu proyecto. ¿Qué es lo que va a hacer diferente a tu aplicación? ¿Qué tipo de público esperas atraer? Etcétera. Los métodos descritos aquí son útiles para capturar el contexto más amplio de tu proyecto de una manera visual y lógica.

Personas (Personajes)

Los personajes (a veces también se utiliza el término personas, como ocurre en inglés) son perfiles ficticios que representan a la audiencia de tu aplicación. La estructura del perfil de un personaje por lo general contiene información como el nombre, la edad y la profesión. También una descripción de sus intereses específicos y de qué manera una propuesta particular encaja en su estilo de vida. ¿Cómo le ayuda a lograr ciertas tareas? En otras palabras, ¿qué motivaría a estos usuarios para darle una oportunidad a tu aplicación?

Normalmente, un proyecto tiene más de un personaje. Lo ideal sería que los perfiles provinieran de investigaciones anteriores, pero incluso si se basan únicamente en tus suposiciones, ponerlos por escrito es uno de los pasos más importantes que puedes llevar a cabo. Para cada personaje puedes seleccionar una imagen de perfil para tener alguna referencia visual. Esto ayuda a generar empatía hacia los usuarios que este personaje representa.

¿Por qué los necesitas? Bueno, no es raro que los fabricantes de un producto se acerquen a la resolución de problemas basándose en sus propias experiencias, preferencias y conocimientos previos sobre un determinado tema. Reconocer diferentes necesidades de usuarios de manera temprana te pone en buen camino para evitar esta trampa. Ante cualquiera paso que tomes a partir de aquí, piensa en cómo percibirán el diseño los usuarios cuando lo vean por primera vez.

Historias de Usuario

Una vez que tienes una mejor idea del tipo de usuarios para los que estás diseñando, puedes comenzar a definir escenarios más detallados: historias de usuario. Pensar en historias de usuario es pensar en flujos: el trayecto desde el punto A hasta el punto B dentro de la aplicación. Ejemplo de historias serían un flujo de compra, subir o compartir una foto, tomar una nota, etc.

Enumerando las historias imprescindibles desde el principio obtendrás una idea de la cantidad de trabajo que queda por delante antes de que la aplicación esté lista para ser publicada. Esto te permitirá capturar la experiencia completa antes de sumergirte en los detalles de la interfaz. Una lista de las historias supone un resumen de tu aplicación, una captura de las principales necesidades de los usuarios. Juntando esto con el conocimiento de los perfiles de tus personajes puedes comenzar ya a avanzar con los detalles del diseño.

Después de tener los principales trayectos descritos, puedes descomponer cada uno de ellos en pasos más pequeños, en tareas. Cada tarea es un pequeño flujo separado dentro de un trayecto más grande. Por ejemplo, para el flujo de subir una foto piensa en aspectos como... ¿necesitarán los usuarios acceder a una galería en su teléfono? ¿Necesitarán seleccionar una imagen? ¿Pueden añadir más de una al mismo tiempo? Todas estas actividades contribuyen a la historia general.

Cuando pienses en una solución a un problema particular sigue regresando a tus perfiles de personajes. Algunos usuarios necesitarán más orientación que otros y puede que quieras considerar añadir algunos mensajes adicionales en la interfaz de usuario para ellos. Incluso aunque es poco probable que todo el mundo se dé cuenta de todos los detalles de información que facilitas, si esta información está disponible para aquellos que la necesitan y se muestra equilibrada de manera que no se interponga en el camino de los usuarios más independientes, el resultado funcionará para todo el grupo en sentido amplio.

Diagrama de Flujo

Un diagrama de flujo muestra los pasos en el flujo y las grandes decisiones que los usuarios van a tomar mientras navegan dentro de tu aplicación. Está definido de manera más lógica y te permite pensar a través del trayecto de los usuarios de una manera visual.

Puedes construir un diagrama de flujo para toda la aplicación o hacer zoom y centrarte en un trayecto en particular. En un nivel superior, un diagrama de flujo describe la estructura de aplicación. Por ejemplo, puedes identificar el número de pantallas diferentes que tendrás que diseñar. Cuando ya te enfocas en un trayecto específico supone una gran técnica para capturar todos los casos para los que tendrás que diseñar (por ejemplo, ¿has pensado en lo que vas a mostrar a los usuarios

si no hay conexión a internet?). A medida que tu proyecto va madurando, puedes iterar en tus diagramas y actualizarlos para que reflejen el estado exacto de tu diseño.

Puedes leer más sobre cómo construir estos diagramas de flujo en Wikipedia[1].

Mapa de Experiencia

Como su propio nombre indica, un mapa de experiencia es una impresión general de toda la experiencia que vas a crear para tus usuarios. Este tipo de resumen general se suele hacer en forma de una infografía/cartel que otras personas involucradas en tu proyecto pueden comprender y relacionar con su contribución al proyecto. Para ilustrar los pasos en la experiencia del usuario puedes utilizar capturas de pantalla de la aplicación o, si todavía se está en las primeras fases o si se desea mantener el mapa más independiente de la implementación, simples ilustraciones o texto.

Dependiendo de la idea, tu aplicación puede suponer toda la experiencia del usuario o puede ser solo uno de los puntos de contacto en el mapa junto con el centro de servicio al cliente, un proveedor, el servicio de entrega, el sitio web, los canales de redes sociales o cualquier otra cosa que encaje en una configuración de tu idea específica.

Si tu aplicación es solo una parte de una experiencia más amplia, dibujar el mapa supone un excelente ejercicio para entender y demostrar el contexto más amplio en el que la aplicación encajará. Cuando entres en la fase de definición de la interfaz de usuario y de diseño de la interacción, tener esos puntos de contacto identificados es realmente útil. La

1 en.wikipedia.org/wiki/Flowchart

comprensión de estas dependencias servirá para decidir de qué manera crear una experiencia consistente en enviar a los usuarios fuera de tu aplicación o darles la bienvenida de nuevo. Los detalles del mapa pueden variar para diferentes personajes y también pueden cambiar dependiendo de si se muestra una experiencia inicial de primera vez o un trayecto de vuelta.

Interfaz de Usuario (UI)

Cuando entiendes de qué manera los trayectos individuales encajan juntos en la visión más global, puedes centrarte en un diseño más definido de la interfaz de usuario. Las técnicas descritas en este apartado son útiles para definir el diseño con suficiente detalle para que se parezca al producto final. Incluso una versión incompleta de la interfaz de usuario será inmediatamente reconocida por los usuarios como una representación de la interfaz, cosa que te permitirá pedirles feedback de manera temprana.

Esta dinámica de pasar de la visión general a los detalles y de regreso a la visión general es en sí misma un gran ejemplo de la naturaleza iterativa del diseño de software. Con cada ciclo obtienes más confianza en que el concepto general tiene sentido (en caso contrario, puedes ajustar el proceso como y cuando sea necesario) y puedes pasar a aplicar la siguiente capa de detalle hasta que todas las piezas estén en su lugar.

Bosquejos (Sketches), Wireframes y Maquetas (Mockups)

Para transmitir diversas ideas de interfaz de usuario puedes simplemente hacer unos bocetos en papel o puedes utilizar una de las muchas herramientas digitales disponibles para dibujar un wireframe. Los wireframes son diseños en baja fidelidad de las pantallas de tu aplicación. Muestran dónde se colocará cada elemento y lo importante que será en relación con los

otros objetos en la misma página. Estas composiciones básicas permiten visualizar cómo de complejos serán los diseños y las interacciones de las pantallas de tu aplicación.

Los wireframes actualizados con frecuencia son un excelente punto de referencia para discutir detalles y próximos pasos a seguir con todos los miembros del equipo. Y si trabajas por tu cuenta, te ayudan a organizar tu propio flujo de trabajo. Si necesitas identificar bibliotecas de patrones de interfaz de usuario en una etapa posterior, podrás guiarte por los elementos que se repiten en tus bocetos.

Así pues, ¿qué nivel de detalle deberían tener tus wireframes? Los bosquejos son ideales en la primera etapa para la generación rápida de ideas. Las aplicaciones más especializadas de creación de wireframes vienen con bibliotecas de widgets preparados para ser organizados rápidamente en una pantalla. La ventaja de utilizar herramientas digitales es que dispones de una versión editable de tus pantallas que después se pueden transformar en prototipos en los que se puede hacer clic. Aunque también puede utilizar bocetos para hacer eso, una versión digital es más fácil de actualizar y mantener cuando trabajas con problemas más detallados.

El concepto de maquetas se utiliza normalmente en referencia a un wireframe con un mayor nivel de detalle en el diseño visual. Algunas maquetas pueden verse idénticas a la aplicación final. Esta visualización más refinada del producto final puede ser útil para demostraciones, aunque a veces un boceto rápido es suficiente para que un equipo altamente colaborativo se ponga de acuerdo sobre los próximos pasos. Está en tus manos decidir qué parte de la apariencia final necesitas en cada paso para avanzar.

Mensajes

Tan pronto como empiezas a usar etiquetas y títulos con sentido, en lugar de los textos por defecto, comienzas a definir la manera de comunicarte con tus usuarios. Dependiendo de la aplicación, el lenguaje tendrá un papel diferente en la orientación de los usuarios a través de los flujos. Pero incluso si el uso de palabras en tu interfaz de usuario es mínimo, no lo dejes en un estado transitorio durante demasiado tiempo.

La redacción es algo que deberías someter a pruebas de usuario. Una simple palabra engañosa podría confundir a tus usuarios o conducirlos a asumir cosas que podrían ir en contra de lo que estás tratando de lograr.

Prototipos

Un prototipo interactivo es la mejor manera de visualizar y evaluar las interacciones de tu aplicación. Es lo suficientemente utilizable como para comunicar el diseño, por lo que no necesitas proporcionar tanta documentación como necesitarías para explicar imágenes estáticas. Tu prototipo puede tener el diseño visual aplicado y verse exactamente como lo hará después de la fase de implementación, o puedes quedarte a nivel de wireframes y centrarte en la funcionalidad y el contenido.

No importa si tienes un gran presupuesto o si estás trabajando en un proyecto personal los fines de semana: tener un prototipo razonablemente completo de tu aplicación es la mejor manera de comunicar su concepto y discutirlo con otros. La narrativa no lineal de tus aplicaciones debe explicarse por sí misma en esta etapa. Muchas de las herramientas de creación de prototipos te permiten experimentarlos en un dispositivo real. Aprovecha esta posibilidad.

Los prototipos normalmente se desarrollan antes de gastar tiempo programando el código y creando un diseño perfecto al nivel del píxel. Un tutorial acordado en el que se pueda hacer

clic es una referencia útil sobre la que los equipos pueden trabajar sin correr el riesgo de salirse demasiado de la línea. También es ideal para probar los prototipos con usuarios y obtener feedback externo.

A la hora de crear un prototipo no hay una única solución buena. Puedes utilizar cualquier técnica que te funcione bien. Desde el prototipado en papel hasta la utilización de alguna de las herramientas especializadas que tienen la funcionalidad para poder hacer clic y navegar por el prototipo. Si tienes habilidades en programación, otra buena posibilidad pasa por construir un prototipo HTML. También puedes crear rápidamente prototipos de una aplicación existente, todo depende de qué enfoque funcione para la configuración de un proyecto específico. En este sentido, cualquier cosa puede ser vista como un prototipo hasta que la aplicación se publica.

Herramientas para Prototipos y Wireframes

Algunas herramientas disponibles son gratis y la mayoría de las comerciales ofrecen versión de prueba o tienen la opción de crear una cuenta gratis para un número limitado de proyectos. Aquí está una lista de algunas aplicaciones que puedes probar:

- **Axure**: *axure.com*
- **Balsamiq**: *balsamiq.com*
- **Framer**: *framerjs.com*
- **Mockingbird**: *gomockingbird.com*
- **OmniGraffle**: *omnigroup.com/omnigraffle*
- **Origami** from Facebook: *facebook.github.io/origami*
- **Penci**: *pencil.evolus.vn*
- **POP**: *popapp.in*
- **Principle**: *principleformac.com*
- **Proto.io**: *proto.io*
- **Sketch**: *sketchapp.com*

Interacciones y Animaciones

Cuando comiences a trabajar con una herramienta de prototipado podrás visualizar la forma en que los usuarios interactuarán con tu aplicación de una manera muy directa. Al poner el prototipo en un dispositivo realmente puedes experimentar cómo se percibirá tu aplicación. Esta experiencia es muy válida no solo de cara a las pruebas con usuarios, sino que muy a menudo tú mismo serás capaz de identificar los elementos que faltan.

La forma en que tu interfaz de usuario se mueve y responde a acciones de los usuarios contribuirá al grado de usabilidad de tu software. Piensa en tus opciones de diseño y en la selección de widgets en función de la plataforma en la que se ejecutará la aplicación. Cada plataforma tiene sus propias convenciones de estilo y métodos específicos para el manejo de las interacciones. Si sigues las prácticas recomendadas, harás que interactuar con tu aplicación inmediatamente resulte más fácil para los usuarios que ya están familiarizados con los patrones de funcionamiento de su dispositivo. Para obtener más información y enlaces a recursos en línea específicos, consulta los capítulos relacionados con plataformas en esta guía.

Con un mercado de dispositivos móviles en constante cambio, también debes considerar la forma en que tu interfaz de usuario se verá en diferentes tamaños de pantalla. Si bien puede ser demasiado pronto para entrar en demasiados detalles antes de haber refinado tu concepto, pensar en la relación diseño-escalabilidad durante la fase de creación de wireframes y de diseño visual (o sea, cuando ya tienes algún tipo de representación gráfica de tus diseños) puede ahorrar una gran cantidad de tiempo de desarrollo y pruebas más adelante. Si este tema te resulta completamente nuevo, merece la pena que leas sobre las mejores prácticas en diseño web adaptativo (RWD, del inglés Responsive Web Design). Los diseñadores

web llevan tiempo resolviendo el problema del escalado de sus diseños. Comprueba también si las directrices específicas de la plataforma proporcionan más información sobre este tema.

Diseño Visual

A menos que estés construyendo una aplicación que utilice un sistema de entrada y salida de información no visual, la interfaz de usuario de tu aplicación va a depender de aspectos gráficos. El cuidado de los detalles en el diseño visual ayudará a mejorar la experiencia de tu aplicación y hacer que se destaque entre las masas.

A estas alturas ya habrás aplicado una serie de principios de diseño gráfico durante la fase de creación de wireframes. Estos incluyen elementos de diseño como el espaciamiento y la jerarquía visual. Un diseño visual pulido no solo mejorará el atractivo estético de tu interfaz de usuario, sino que una estética bien ejecutada mejorará la funcionalidad de tu aplicación y reducirá la curva de aprendizaje para los usuarios, proporcionándoles indicaciones visuales.

La coherencia de estilo a través del flujo ayuda a tus usuarios a entender tu interfaz de usuario y a aprender más rápido cómo funcionan sus interacciones. Por ejemplo, considera cómo afectará a tus usuarios si el botón de la acción principal cambia de color de una pantalla a otra. ¿Van a estar confundidos? ¿Entenderán la razón del cambio? Si las modificaciones de estilo son intencionales asegúrate de que las estás haciendo por razones de usabilidad.

Al igual que en el diseño de pantallas e interacciones, en el nivel de prototipado algunas decisiones de estilo pueden venir dadas por las directrices específicas de la plataforma. Tu aplicación puede verse muy diferente dependiendo de la plataforma para la que se definió. Asegúrate de que tu diseño sigue las prácticas recomendadas para el uso de fuentes, iconos

estándar y convenciones de diseño. Una vez más, consulta los capítulos relacionados con las plataformas en esta guía para encontrar más información y enlaces a recursos específicos.

Aplicar la imagen corporativa de la empresa en la interfaz de usuario se puede conseguir de una manera no obstructora para que los usuarios puedan concentrarse en la interacción con tu aplicación. Utiliza el fondo, el color de los controles y tal vez algunas imágenes u opciones de diseño para lograr que la imagen de la marca sea reconocible. Una pantalla de presentación (en caso de existir) es un buen lugar para mostrar algunos gráficos adicionales.

Por último, el icono de inicio es el primer elemento visual por el que tu aplicación va a ser identificada y juzgada. Haz que se vea bien. Si estás planeando publicar en múltiples plataformas, comprueba los requisitos de diseño al principio del proyecto para que puedas trabajar con gráficos compatibles entre plataformas.

El diseño visual es un área muy rica. Si quieres entender más acerca de sus técnicas específicas puedes encontrar más información y ejemplos en línea (consulta algunos sitios web recomendados al final).

Pruebas de Usuarios

La mejor manera de validar tu concepto de interfaz es mostrársela a los usuarios tan pronto como tu trabajo sea lo suficientemente representativo como para solicitar feedback. No tienes que esperar hasta tener un producto acabado y pulido. Hacer pruebas pronto puede ahorrarte mucho tiempo a largo plazo. Expondrá conceptos que no funcionan al principio del proceso. Cuanto más tiempo inviertas en el desarrollo de tus diseños, más difícil te será abandonarlos y comenzar de nuevo. Es más difícil aceptar feedback sobre algo que consid-

eras casi terminado que sobre un prototipo en el que se pueda hacer clic y que puedas actualizar rápidamente.

Una sesión típica de pruebas de usuario dura aproximadamente una hora. Durante ese tiempo se pide a usuarios no familiarizados con el producto que lleven a cabo ciertas tareas, por lo general sobre funcionalidad básica. Durante la búsqueda de personas para entrevistar es bueno referirse a las descripciones originales de personajes y buscar usuarios que respondan a esos perfiles.

Para hacer el mejor uso del tiempo de pruebas, prepáralas con antelación. Anota las presentaciones, piensa en cómo vas a explicar la dinámica de la sesión a los usuarios y cómo vas a utilizar sus comentarios. También debes preparar las tareas que se corresponden con lo que deseas probar. Prepara una lista con ellas y ten a mano tus notas para asegurarse de que no te olvidas de pedirlas.

Es bueno mencionar a los usuarios que el prototipo no está completo y que puede haber algunas partes inacabadas. Si asumen que la persona que ejecuta la sesión es el autor del diseño, podrían sentirse cautelosos a la hora de dar feedback crítico. Asegúrales que son libres de expresar sus opiniones honestamente. Después de todo, la única razón por la que has organizado la sesión de pruebas es para obtener un feedback independiente. Una vez que se han fijado las expectativas, es importante que sigas las reglas y no encamines a los usuarios a ninguna conclusión. No les ayudes explicándoles cómo funcionan las cosas (a menos que no lo puedan entender y ello suponga que no puedas continuar con la sesión) y formula tus preguntas de una forma no interruptora. Durante la sesión graba el feedback de los usuarios o asegúrate de tomar suficientes notas.

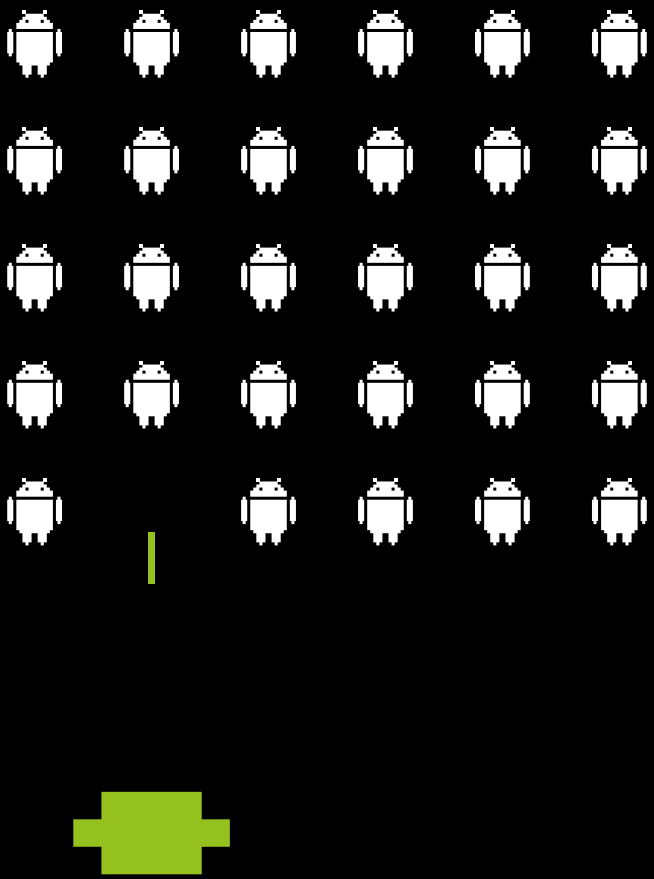
Cuando obtienes feedback, puedes iterar sobre tu diseño y mejorar las partes que no estaban del todo correctas o, si el feedback ha sido positivo, seguir adelante hacia la siguiente fase de desarrollo.

Incluso si no puedes probar con un gran número de personas, hacer pruebas con solo algunos usuarios desvelará los principales problemas que son los más propensos a causar problemas de usabilidad. Una única opinión no sesgada es mejor que ninguna opinión en absoluto. Si todavía estás explorando nuevas áreas y tu propio prototipo aún no está listo, puedes ejecutar sesiones de pruebas sobre otras aplicaciones que ya hayan sido publicadas. Puede llegar a sorprenderte las muchas cosas en las que tú nunca has caído que otros pueden notar acerca de una aplicación.

Aprender Más

Hay un montón de recursos disponibles en línea. Estos son algunos para abrirte el apetito:

- **UX Archive**: *uxarchive.com*
- **User Onboarding**: *useronboard.com*
- **Smashing Magazine**: (sección de diseño de UX): *uxdesign.smashingmagazine.com*
- **UX Magazine**: *uxmag.com*
- **UX Matters**: *uxmatters.com*
- **Nielsen Norman Group**: *nngroup.com*
- **Interaction Design Foundation**: *interaction-design.org*



Android

El Ecosistema

La plataforma Android la desarrolla la Open Handset Alliance, liderada por Google, y ha estado a disposición del público desde noviembre del 2007. Su uso por parte de la mayoría de fabricantes de hardware la ha convertido en el sistema operativo de smartphones con el mayor crecimiento de todos los tiempos y el que domina el mercado a día de hoy: más del 82% de todos los smartphones vendidos en el segundo trimestre del 2015 en todo el mundo utilizaban Android[1], el 71% de los desarrolladores para móviles lo hacen para Android[2]. En septiembre del 2015, Google anunció que más de 1.400 millones de dispositivos Android han sido activados hasta la fecha[3] incluyendo wearables, tablets, reproductores multimedia, decodificadores de televisión, teléfonos de sobremesa y sistemas de entretenimiento para automóviles. Las propias gafas inteligentes de Google, las Google Glass, también utilizan una versión reducida de Android compatible tanto con aplicaciones web como nativas. El número de aplicaciones Android en la Google Play ha sobrepasado los 1,8 millones en diciembre del 2015[4].

1 idc.com/prodserv/smartphone-os-market-share.jsp

2 a veces entre otros, revisa el informe Developer Economics Report disponible vía developereconomics.com

3 androidcentral.com/google-says-there-are-now-14-billion-active-android-devices-worldwide

4 appbrain.com/stats/number-of-android-apps

Android es un sistema operativo, una colección de aplicaciones preinstaladas y una plataforma de desarrollo de aplicaciones soportada por un amplio paquete de herramientas. La plataforma sigue evolucionando rápidamente con adición regular de nuevas características aproximadamente cada 6 meses. La última versión es Android 6.0 'Marshmallow', que introdujo nuevas características como un nuevo sistema de permisos y de gestión de energía y que mejora las principales características de su predecesor L 'Lollipop'. La mejora más importante de Lollipop había sido la introducción de un nuevo conjunto de herramientas para la interfaz de usuario llamado Material Design[5] que tiene como meta unificar todas las plataformas como teléfonos, wearables y televisores bajo una misma filosofía de diseño.

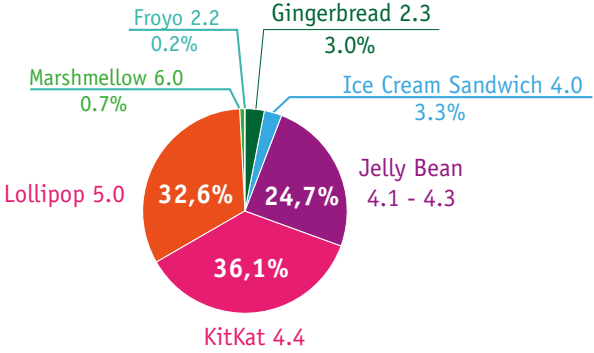
Otra mejora importante en Android L fue la inclusión del nuevo Android Runtime (ART) que proporciona características como una recolección de basura (garbage collection o GC) más inteligente que llega a cuadriplicar el rendimiento. El proyecto Volta introdujo herramientas para mejorar la duración de la batería optimizando el comportamiento de la aplicación. Otras funciones incluyen control de privacidad, compartición de cuentas entre dispositivos y extensiones al sistema de notificaciones.

5 developer.android.com/design/material/index.html



Uno de los problemas más debatidos cuando se desarrolla para Android es la fragmentación del sistema: la multitud de dispositivos diferentes de diversos fabricantes y el rápido progreso de la plataforma en sí mismo generan incertidumbre sobre si una aplicación Android podrá ejecutarse o no en cualquier dispositivo. Además, la adopción de la última versión del sistema operativo está siendo más lenta que en otras plataformas. En cualquier caso, actualmente puedes alcanzar más del 90% de la base instalada si decides centrarte en

Android 4.0 o versiones superiores[6].



(datos de enero del 2015)

6 developer.android.com/about/dashboards

Android Marshmallow

Android 6.0 "Marshmallow" es la versión más reciente del sistema operativo Android, anunciado en la conferencia Google I/O en mayo del 2015 con el nombre en clave "Android 'M'". Ha sido lanzado oficialmente en octubre del 2015.

Uno de los cambios más significativos en Marshmallow es la nueva arquitectura de permisos. Los usuarios ya no tienen que otorgar todos los permisos que una aplicación requiere durante la instalación. La aplicación ahora puede (y debería) solicitar permisos a medida que vayan siendo necesarios. Se anima a los desarrolladores a seguir las mejores prácticas actuales consistentes en pedir permiso solo en el momento en que es necesario, dando al usuario tanta información de contexto como sea posible.

Marshmallow también introduce nuevas APIs para los asistentes contextuales (como "Google Now On Tap"), un nuevo sistema de administración de energía que reduce la actividad en segundo plano cuando un dispositivo no está siendo manejado físicamente, soporte nativo para el reconocimiento de huellas digitales y conectores USB Type-C, y la capacidad de mover datos a un almacenamiento extraíble (por ejemplo, una tarjeta microSD).

Material Design

Durante la conferencia Google I/O en junio del 2014, Google dio a conocer su nuevo lenguaje de diseño basado en papel y tinta, llamado Material Design. Originalmente con el nombre en clave de Quantum Paper, Material Design extiende el concepto de "tarjeta" visto anteriormente en Google Now. Dice el diseñador Matías Duarte que "a diferencia del papel de verdad, nuestro material digital se puede ampliar y reformar de forma inteligente. Material tiene superficies físicas y bordes. Las

junturas y sombras proporcionan información sobre lo que se puede tocar".

Quizás por primera vez, Material Design trae al ecosistema Android una fuerte y consistente identidad visual, paralela pero distinta del diseño plano de iOS y del diseño Metro de Windows. El lenguaje visual en sí mismo está bien especificado y documentado.[7]. Para fomentar una experiencia de usuario sólida y una apariencia consistente en las aplicaciones Android, Google publica una guía de diseño[8]. Al introducirnos en la importancia de los esquemas de color, pautas de diseño y el nuevo diseño Material, la guía proporciona una orientación excelente a la hora de construir aplicaciones para el ecosistema Android.

También hay una aplicación canónica de Material Design para las interfaces de usuario de aplicaciones web llamada Polymer Paper Elements[9].

Android Wear

Android Wear[10] fue anunciado por Google en marzo del 2014, en asociación con Motorola, Samsung, LG, HTC y Asus. Se trata básicamente del sistema operativo Android portado a smartwatches y otros wearables, que pueden emparejarse con teléfonos que utilicen Android con versión 4.3 o posterior (hay algo de soporte limitado para el emparejamiento con un iPhone). Los wearables integran Google Now y la tienda de Google Play.

Una característica clave es el ecosistema de aplicaciones de Google Fit que permiten cosas como monitorizar sesiones

7 google.com/design/spec/material-design

8 developer.android.com/design

9 elements.polymer-project.org

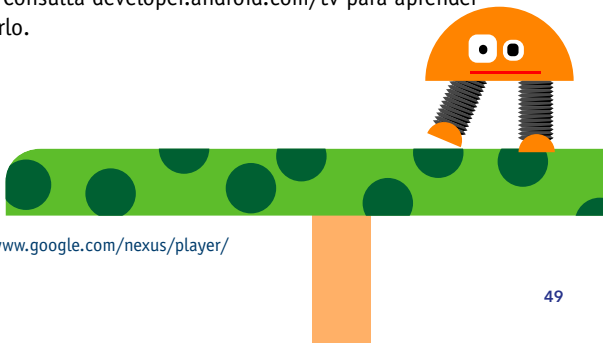
10 developer.android.com/training/building-wearables.html

de correr, seguir la actividad del corazón, contar el número de pasos, etc. Los usuarios pueden utilizar su reloj para controlar su teléfono (por ejemplo la música). También pueden recibir notificaciones a través del motor de vibración, tanto para notificaciones recibidas desde Google Now como para recordatorios de vuelos, avisos de tráfico, recordatorios de reuniones, etc.

Android TV

Anunciado en la conferencia Google I/O en junio del 2014, {Android TV android.com/tv} estaba basado en Android 5.0 "Lollipop" y es un sucesor de la anterior tentativa de Google sobre televisión inteligente (Google TV). Android TV está diseñado para ser integrado tanto en televisores como en reproductores de medios digitales. Google ha desarrollado el primer dispositivo Android TV con Asus, el Nexus Player[11], lanzado en noviembre del 2014. Varios fabricantes de televisores, como Sony, Sharp y Philips, han integrado ya Android TV en sus pantallas a día de hoy. Los usuarios pueden descargar aplicaciones y juegos desde la Play Store integrada, incluyendo aplicaciones de medios como YouTube, Hulu y Netflix.

Las aplicaciones de Android TV utilizan la misma estructura que las de los teléfonos y tablets. De este modo los desarrolladores pueden aprovechar tanto sus aplicaciones como su conocimiento ya existentes para enfocarse en la plataforma de televisión. Consulta developer.android.com/tv para aprender cómo hacerlo.



11 https://www.google.com/nexus/player/

Primeros Pasos

El lenguaje de programación principal para Android está basado en Java. Pero cuidado, solo un subconjunto de las librerías y paquetes de Java son compatibles y hay muchas APIs específicas de la plataforma que no funcionarán en Android. Puedes encontrar respuestas a tus preguntas "¿Qué y Por qué?" en línea en la Dev Guide para Android[12] y a las preguntas "¿Cómo?" en la documentación de referencia[13]. Además, Google ha introducido una sección en su documentación llamada "Android Training"[14] que ayuda a los nuevos desarrolladores a aprender varias de las mejores prácticas. Aquí es donde puedes aprender sobre conceptos básicos como la navegación y comunicación entre aplicaciones, así como funciones más avanzadas como la descarga inteligente de bitmaps y la optimización de la aplicación para reducir el consumo de la batería.

Para empezar, necesitas el SDK de Android[15], que está disponible para Windows, Mac OS X y Linux. Contiene las herramientas necesarias para crear, probar, depurar y analizar aplicaciones. Las herramientas de desarrollo de Android (ADT)[16] son responsables de la integración con IDEs y de asegurarse de que tu flujo de desarrollo sea lo más cómodo posible.

Soporte de IDEs

A día de hoy, Google ofrece un IDE basado en IntelliJ llamado "Android Studio" y Eclipse (entendido como "Eclipse + ADT Plugin"), uniendo de manera eficaz las Android Developer

12 developer.android.com/guide
13 developer.android.com/reference
14 developer.android.com/training/index.html
15 developer.android.com/sdk
16 developer.android.com/tools/sdk/eclipse-adt.html

Tools con el IDE. Android Studio[17] es ahora el IDE oficial para Android y viene directamente con soporte para Gradle y muchas funciones especialmente diseñadas para el desarrollo Android.

IDE	soporte a plugin	versión incluida
Eclipse	paquete ADT por separado	Eclipse + ADT Plugin
IntelliJ	plugin Android por separado	Android Studio

Encontrarás más información y las descargas necesarias en la sección "Tools" de la documentación de Android[18].

Desarrollo Nativo

El Android NDK[19] permite la creación de componentes nativos para tu aplicación al aprovechar tanto JNI para la invocación de métodos nativos como el uso de subclases nativas que ofrecen callbacks a aquellos no nativos. Esto es importante tanto para desarrolladores de videojuegos como para cualquiera que necesite procesamiento fiable.

Implementación

Arquitectura de la Aplicación

Las aplicaciones Android incluyen habitualmente una mezcla de Activities, Services, BroadcastReceivers y ContentProviders, todos los cuales deben estar declarados en el manifiesto de la

17 http://developer.android.com/sdk/index.html

18 developer.android.com/tools

19 developer.android.com/tools/sdk/ndk

aplicación. El manifiesto también incluye los metadatos de la aplicación, como el título, la versión y los permisos requeridos.

Una Activity es un bloque de funcionalidad con una interfaz de usuario adjunta. Un Service se utiliza para tareas que se ejecutan en segundo plano y, por lo tanto, no están ligadas directamente a una representación visual. Un BroadcastReceiver gestiona los mensajes transmitidos por el sistema, por ti mismo o por otras aplicaciones. Un ContentProvider es una interfaz para el contenido de una aplicación que abstrae de los mecanismos de almacenamiento subyacentes (por ejemplo, SQLite).

Una aplicación puede estar formada por varios de estos componentes, por ejemplo, una Activity para la interfaz de usuario y un Service para tareas cuya ejecución se alarga. La comunicación entre los componentes se consigue mediante Intents o llamadas a procedimientos remotos vía Android Interface Definition Language (AIDL).

Los Intents empaquetan datos, tales como la localización del usuario o una URL, con una acción y desencadenan acciones en la plataforma, pudiendo ser utilizados como un sistema de mensajería en tu aplicación. Por ejemplo, el Intent para mostrar una página web abrirá el navegador. La fuerza de esta filosofía basada en bloques de construcción es que se puede sustituir cualquier funcionalidad por otra aplicación, ya que Android siempre utiliza la aplicación preferida para cada Intent específico. Por ejemplo, el Intent para compartir una página web lanzado por una aplicación de lectura de noticias puede abrir un cliente de correo electrónico o una aplicación de mensajería de texto, en función de las aplicaciones instaladas y de las preferencias de usuario (se puede utilizar

cualquier aplicación que declare que puede responder al Intent de compartir).

La interfaz de usuario de una aplicación está separada del código en archivos XML de diseño específicos de Android. Se pueden crear diferentes diseños para diferentes tamaños de pantalla, localizaciones por países y características del dispositivo sin tener que tocar el código Java. Con este fin, textos e imágenes están organizados en carpetas de recursos separadas. Evidentemente también puedes definir y crear diseños vía código o hacer uso de ambas estrategias para permitir actualizaciones dinámicas de la interfaz de usuario.

El SDK y los Plugins

Para ayudarte en el desarrollo tienes muchas herramientas a tu disposición en el SDK. Las más importantes son las siguientes:

- **android:** para crear un proyecto o gestionar dispositivos virtuales y versiones del SDK.
- **adb:** para llamar a los dispositivos, conectarse e interactuar con ellos (también con dispositivos virtuales) moviendo archivos, instalando aplicaciones, etc.
- **emulator:** para emular las características de un dispositivo virtual. Lleva un tiempo arrancarlo, por lo que conviene hacerlo solo una vez por sesión de trabajo y no para cada compilación.
- **ddms:** para observar el interior del dispositivo o emulador, ver los mensajes del registro (log) y controlar funciones del emulador como la latencia de red y la posición GPS. También se puede utilizar para monitorizar el consumo de memoria y terminar procesos. Si esta herramienta está en ejecución, también puedes conectar el depurador de Eclipse a un proceso que se esté ejecutando en el emulador. Aparte, ddms es la única manera (sin acceso

root) para realizar capturas de pantalla en versiones de Android anteriores a la 4.0.

Estas cuatro herramientas y muchas otras se pueden encontrar en el directorio de herramientas del SDK, incluyendo algunas para el análisis de los registros de rastreo de métodos, para la inspección de diseños y para la prueba de aplicaciones con eventos al azar. Si estás experimentando problemas, como por ejemplo la generación de excepciones, asegúrate de revisar el registro del ddms o utilizar el mecanismo logcat.

Si estás utilizando características como Fragments[20] para pantallas grandes, asegúrate de agregar el paquete Android Compatibility de Google. Está disponible a través del SDK y AVD Manager y permite desarrollar para versiones anteriores de Android utilizando prestaciones modernas. Asegúrate también de utilizar los paquetes v4 en tu aplicación para proveer de máximo soporte a versiones previas. También hay una versión para Android 2.1 y versiones superiores, la librería v7 appcompat, que ofrece una manera de implementar el patrón ActionBar entre otras cosas, tal como se explica en la documentación en línea[21].

Desarrollar tu aplicación para Android 3.1+ te permitirá crear widgets redimensionables para la pantalla de inicio y conectar por USB a varios dispositivos, como cámaras digitales y gamepads, entre otros. Las versiones Android 4.X introdujeron interesantes novedades, como notificaciones expandibles, widgets de bloqueo de pantalla y una cámara con detección de rostros. El Material Design UI Toolkit fue introducido con Android 5.0 e incluye nuevos widgets y otros elementos para usar en teléfonos, wearables y otras plataformas. El framework

20 developer.android.com/guide/topics/fundamentals/fragments.html

21 developer.android.com/tools/support-library/features.html

nativo Renderscript (introducido en la 3.1) ha cambiado mucho y ya no proporciona la capacidad directa de renderizar gráficos, sino que ahora se puede utilizar para tareas exigentes en procesamiento.

Para ofrecer compatibilidad con dispositivos con versiones anteriores de Android, Google comenzó a utilizar el framework Google Play Services[22], que se actualiza a través de la Play Store y añade librerías, como por ejemplo la más reciente de Google Maps. Si estás interesado en autenticar usuarios, puede que te interese echar un vistazo a los beneficios que aporta tener datos reales de usuarios en tu aplicación con Google+ Sign. La funcionalidad se gestiona vía tokens de OAuth 2.0 que permiten el uso de la cuenta Google del usuario en su nombre.

Testeo

El primer paso para probar una aplicación es ejecutarla en el emulador o en un dispositivo. A partir de ese momento es posible depurarla, si es necesario, a través de la herramienta ddms.

Todas las versiones del sistema operativo Android están preparadas para ejecutarse en dispositivos sin necesidad de modificaciones, sin embargo algunos fabricantes de hardware pueden haber realizado cambios en algunos elementos de la plataforma. Por tanto, es esencial probar la aplicación en varios dispositivos. La lista de AppBrain[23] puede ayudarte a hacerte una idea de qué dispositivos son los más populares.

Para automatizar las pruebas, el SDK de Android viene con

22 developer.android.com/google/play-services/

23 www.appbrain.com/stats/top-android-phones

algunas herramientas muy útiles[24]. Los tests se pueden escribir con el formato estándar JUnit, utilizando los objetos simulados de Android (mock objects) que se incluyen en el SDK.

Las clases Instrumentation pueden monitorizar la interfaz de usuario y enviar eventos de sistema tales como pulsaciones de teclas. A partir de aquí, tus pruebas pueden comprobar el estado de tu aplicación tras estos eventos. MonkeyRunner[25] es una herramienta de automatización de pruebas potente y extensible que permite probar toda la aplicación. Estas pruebas se pueden ejecutar tanto en dispositivos virtuales como físicos.

En la revisión 21 del SDK, Google introdujo finalmente un framework de automatización de pruebas de interfaz de usuario más eficiente[26] que permite realizar pruebas funcionales de interfaz de usuario en Android Jelly Bean y versiones superiores. La herramienta puede ser ejecutada desde el shell con el comando `uiautomatorviewer` y muestra la interfaz capturada, incluyendo información sobre las vistas. Ejecutar las pruebas es relativamente fácil: una vez que hayas escrito tu test, se compila a través de ANT como un archivo JAR. Este archivo tiene que ser enviado al dispositivo y ejecutado con el comando `adb shell uiautomator runtest`.

Espresso[27] proporciona una API muy sencilla que ayuda a escribir pruebas procedurales para la interfaz de usuario con rapidez.

24 developer.android.com/guide/topics/testing/testing_android.html

25 developer.android.com/guide/developing/tools/monkeyrunner_concepts.html

26 android-developers.blogspot.de/2012/11/android-sdk-tools-revision-21.html

27 googletesting.blogspot.de/2013/10/espresso-for-android-is-here.html

Algunos frameworks de pruebas de código abierto, como Robotium[28], pueden complementar tus demás tests automatizados. Robotium incluso puede ser utilizado para probar archivos binarios apk si el código fuente de la aplicación no está disponible. Roboelectric[29] es otra gran herramienta que ejecuta las pruebas directamente sobre tu IDE en la JVM estándar de tu ordenador.

Tus pruebas automatizadas se pueden ejecutar en servidores de integración continua como Jenkins o Hudson. Roboelectric se ejecuta en una JVM estándar y no necesita un entorno con el runtime Android. La mayoría de los demás frameworks automatizados de pruebas, incluyendo Robotium, se basan en el framework Instrumentation de Android y requerirán ser ejecutados en la JVM correspondiente. Plugins como el Android Emulator Plugin[30] permiten que estos tests sean configurados y ejecutados en Hudson y Jenkins.

Compilación

Aparte de compilar tu aplicación directamente en tu IDE favorito, hay maneras más flexibles de construir aplicaciones Android. Gradle[31] es actualmente la herramienta de creación automatizada con soporte oficial para Android. También hay un plugin de Maven[32] que cuenta con gran apoyo en la comunidad de desarrolladores. Ambas herramientas pueden utilizar

28 code.google.com/p/robotium

29 robolectric.org/

30 wiki.hudson-ci.org/display/HUDSON/Android+Emulator+Plugin

31 tools.android.com/tech-docs/new-build-system

32 code.google.com/p/maven-android-plugin/

dependencias de diferentes repositorios Maven, como el Maven Central Repository[33].

El sistema que utiliza Google para publicar librerías para Gradle se conoce como Android Archive (archivos .aar) y se puede acceder a él utilizando el Android SDK Manager. También puedes crear tus propias librerías o SDKs utilizando el plugin android-library para Gradle. Una excelente fuente de librerías Android compatibles con Gradle es "Gradle, please"[34].

Firmado

Tus aplicaciones se firman siempre durante el proceso de compilación, ya sea con una firma de depuración o con una de publicación. Puedes utilizar un mecanismo de autofirma, lo que evita gastos relacionados con la firma (y seguridad).

La misma firma debe ser utilizada para las actualizaciones de tus aplicaciones, así que asegúrate de no perder tu archivo keystore o la contraseña. Recuerda: puedes utilizar una única clave para todas tus aplicaciones o crear una nueva para cada una de ellas.

Distribución

Una vez hayas creado tu aplicación rompedora y la hayas probado, debes publicarla en la tienda de aplicaciones de Android llamada "Play" *en play.google.com/apps/publish*.

Estás obligado a inscribirte en el servicio a través de tu cuenta de Google Checkout y pagar una cuota de inscripción de 25 dólares. Una vez aprobado tu registro, puedes cargar la aplicación, añadir imágenes y descripciones y publicarla.

33 www.maven.org

34 gradleplease.appspot.com

Asegúrate de que has definido `versionName`, `versionCode`, un icono y una etiqueta en tu `AndroidManifest.xml`. Además, las características declaradas en el manifiesto (los nodos uses-feature) se utilizan para filtrar aplicaciones para distintos dispositivos.

Una de las novedades más recientes en la Google Play Store es que aceptan tests de tipo alfa y beta y lanzamientos progresivos. Esto te permite realizar pruebas interesantes con usuarios antes de lanzar la aplicación y hacerla pública a todos los usuarios. Aún más, puedes dirigirlas a países y dispositivos específicos utilizando la Developer Console y exportar las estadísticas resultantes para comprender mejor tu base de usuarios. Utilizando los servicios de localización integrados, puedes añadir nuevos idiomas a tu aplicación fácilmente a cambio de un pago (asegúrate de revisar la Localisation Checklist[35] para obtener información detallada acerca de la importancia de este tema).

Como quiera que hay una gran cantidad de aplicaciones compitiendo en Android Play, es posible que desees utilizar tiendas alternativas de aplicaciones[36]. Ofrecen diferentes métodos de pago y pueden dirigirse a grupos específicos de consumidores. Una de esas tiendas es la Amazon Appstore, que viene instalada en la familia de tablets Kindle Fire.

Adaptación

A medida que la adopción de Android se incrementa, también lo hace un ecosistema de elementos específicos de fabricantes que incluyen sus propios SDKs, versiones de Android completamente adaptadas y herramientas en torno a temas tales

35 developer.android.com/distribute/googleplay/publish/localizing.html

36 onepf.org/appstores

como las pruebas de versiones alfa y beta. Esto tiene su lado positivo, por ejemplo una integración sólida que posibilita una fantástica experiencia de usuario, pero también negativo, ya que incrementa la fragmentación del ecosistema. Las tiendas de los fabricantes a menudo bloquean la publicación de aplicaciones genéricas que utilicen servicios de terceros que no sean los suyos propios.

Un ejemplo es el ecosistema de la Kindle Fire de Amazon, que es básicamente una nueva y personalizada rama de Android y supone la tablet Android con mayor cuota de mercado. En vez de utilizar Google Play Services para utilizar Google Maps o la compra de aplicaciones, obliga a utilizar las librerías propias de Amazon que ofrecen funcionalidades similares. Las razones tras esto son sencillas: los dispositivos Kindle no se comercializan con las librerías necesarias para ejecutar servicios de Google. Amazon ofrece también sus propios servicios de juegos y publicidad (comparable a Google Play Games) que te ayudan a dirigirte a tu audiencia. Además, ofreciendo emuladores para sus cuatro dispositivos (1st Gen, 2nd Gen, HD 7" y HD 8.9"), Amazon proporciona un entorno realista para perfeccionar las aplicaciones. Finalmente, Amazon ofrece herramientas de pruebas a su comunidad de desarrolladores y también verifican todas aquellas aplicaciones que se suben a su tienda.

A continuación, un breve resumen que te ayudará a localizar los recursos que puedas necesitar:

Fabricante	Documentación
Amazon	developer.amazon.com/sdk/fire.html
HTC	htcdev.com
LG	developer.lge.com
Motorola	developer.motorolasolutions.com/community/android
Samsung	developer.samsung.com/android
Sony	developer.sonymobile.com

Curiosamente, un número creciente de fabricantes (por ejemplo Samsung y HTC) han comenzado a ofrecer versiones no personalizadas de Android de sus dispositivos, etiquetadas como "Google Play Edition". Estos dispositivos utilizan el mismo hardware que los modelos estándar, pero vienen sin software personalizado. Se distribuyen directamente a través de la Play Store de Google y ofrecen los dispositivos más innovadores y novedosos a los usuarios que quieren adherirse a una experiencia Google más pura.

De manera adicional a las versiones de los principales fabricantes, el Android Open Source Project (AOSP)[37] ofrece una versión de código abierto de la pila Android para crear ROMs personalizadas y portar dispositivos a la plataforma Android. Fabricantes independientes como Fairphone[38] utilizan AOSP para crear su propia versión de la plataforma Android. La desventaja de este enfoque es que se echan en falta servicios

37 source.android.com

38 fairphone.com

de Google como la Google Play Store habitualmente disponible en los principales dispositivos Android.

Monetización

Además de vendiéndola en alguna de las muchas tiendas de aplicaciones, hay muchas maneras diferentes de monetizar una aplicación Android. Una forma viable es mediante el uso de la publicidad, que puede estar basada en clics o en número de visitas, y puede proporcionar un ingreso estable. Además de esto existen diferentes posibilidades para facturar desde la aplicación, como el servicio propio de facturación de Google[39] que utiliza la tienda Google Play, el SDK móvil de PayPal[40] y Mobile Payments Library[41]. La mayoría de los servicios difieren en las comisiones por transacción y en las posibilidades que ofrecen, por ejemplo suscripciones, pagos paralelos o pagos previamente aprobados. Si buscas añadir una funcionalidad muy atractiva a tu aplicación, considera implementar el SDK de card.io[42] para el escaneado de tarjetas de crédito con la cámara del móvil.

Para los ecosistemas específicos de los fabricantes, como Samsung Apps o Amazon Appstore, considera el uso de sus SDKs para beneficiarte de sus optimizados modelos de monetización.

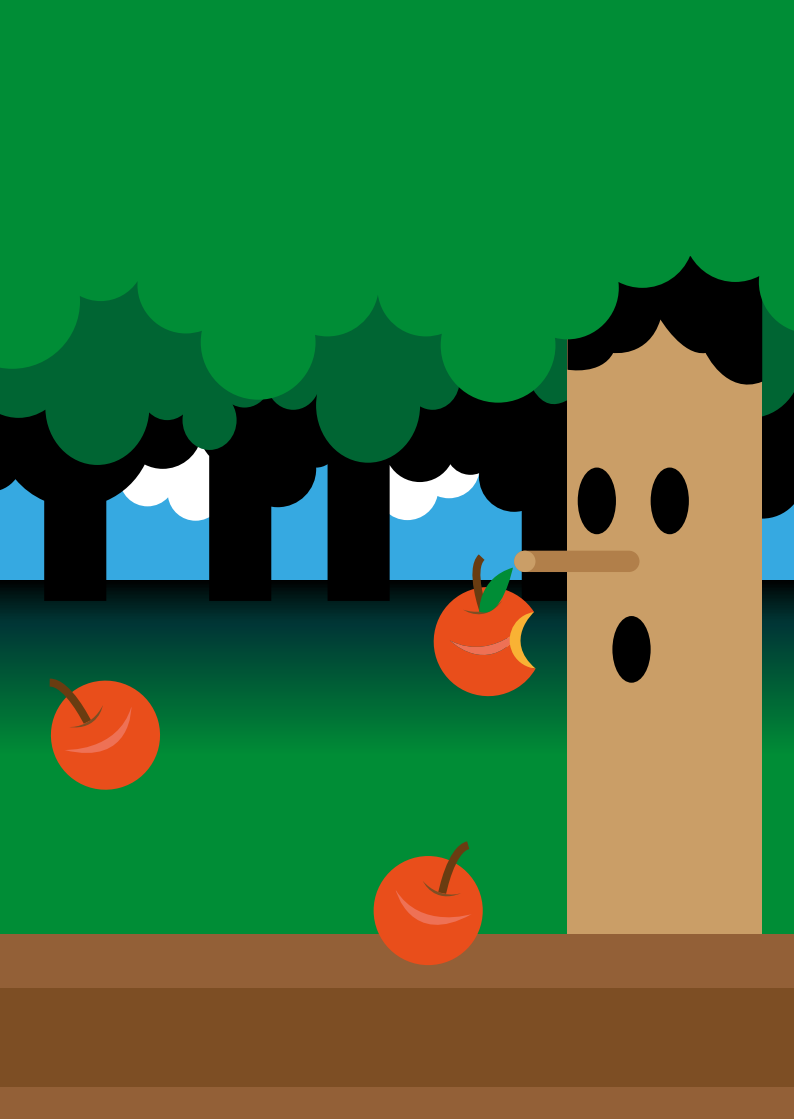
Asegúrate de verificar que el método de pago que elijas está en armonía con los términos y condiciones de las diferentes tiendas en las que deseas publicar la aplicación. Especialmente

39 developer.android.com/google/play/billing/

40 github.com/paypal/PayPal-Android-SDK

41 developer.paypal.com/webapps/developer/docs/classic/mobile/gs_MPL/

42 card.io



BY Alex Repty

iOS

El 9 de enero del 2007, Steve Jobs presentó una nueva categoría de producto para la entonces empresa de ordenadores y dispositivos musicales. El día antes de la demostración, Jobs aún no podía realizarla de inicio a fin en un iPhone (solo existían alrededor de 100 prototipos) sin que este, "rechazara llamadas al azar, perdiera la conexión a internet, se bloqueara o simplemente apagara"[1]. Pero cuando llegó el momento de la presentación, Jobs la realizó magistralmente y el resto es historia.

Con el lanzamiento del primer iPhone, Apple dio a conocer un nuevo sistema operativo para utilizar el dispositivo. Su nombre original era iPhone OS ya que el iPhone fue el único dispositivo que podía ejecutarlo. En noviembre del 2010, coincidiendo con el lanzamiento de la cuarta generación del sistema operativo, Apple le cambió el nombre y pasó a llamarse iOS para coincidir con el lanzamiento del primer iPad. Esta versión fue nombrada iOS 4 y desde entonces se ha lanzado una nueva versión cada año, culminando en la versión actual, iOS 9, que salió al mercado en septiembre del 2015.

Mientras que iOS 7 fue promocionado a partir de su nueva interfaz de usuario, iOS 8 se centró en los nuevos frameworks y servicios, así como en el último modelo de dispositivo, el Apple Watch. También introdujo una mayor integración entre dispositivos iOS y ordenadores Mac con Yosemite, que permite a los usuarios iniciar tareas, como la creación de correos electrónicos, en un dispositivo y acabarlas en otro a través

1 www.nytimes.com/2013/10/06/magazine/and-then-steve-said-let-there-be-an-iphone.html

de un concepto llamado Handoff. iOS 9 mejoró la multitarea e introdujo los enlaces profundos.

El Ecosistema Hoy

Desarrollar para iOS se ha vuelto más popular que nunca. En el tercer trimestre del 2015, el número de aplicaciones iOS sobrepasó el millón y medio[2]. Los ingresos totales de aplicaciones en iOS solo en 2015 ascendieron a 20.000 millones de dólares, lo que significa que los desarrolladores de iOS han recibido un total acumulado de 40.000 millones de dólares del AppStore de Apple[3].

Base Instalada de iOS

Además de vender más de 1.000 millones de dispositivos iOS, un plus a favor de Apple es la alta tasa de adopción de cada versión de iOS poco después de su lanzamiento. Esto permite a los desarrolladores centrarse en desarrollar para la versión más reciente y no preocuparse en dar soporte a una gran cantidad de dispositivos en versiones anteriores, cosa que ha sido un reto para los desarrolladores Android. Doce semanas después del lanzamiento de iOS 9 Mixpanel anunció ya una tasa de adopción de casi el 80% de todos los dispositivos iOS[4] con un 16% todavía en iOS 8, dejando solamente un 4% de dispositivos corriendo versiones anteriores de iOS. Compara esto con la versión 5.0/5.1 Lollipop del sistema operativo de Android, que después de un año todavía no ha sido capaz de captar siquiera

2 statista.com/statistics/276623/number-of-apps-available-in-leading-app-storesapps

3 apple.com/pr/library/2016/01/06Record-Breaking-Holiday-Season-for-the-App-Store.html

4 mixpanel.com/trends/#report/ios_9

un tercio de los dispositivos Android[5]. La última versión lanzada, 6.0 Marshmallow, se ha utilizado solo en un 0,5% de dispositivos Android dos meses después de su lanzamiento.

Dispositivos Ejecutando iOS

En lugar de enumerar todos los dispositivos iOS que Apple ha creado, aquí están los dispositivos actuales que soportan iOS 9, ya que esta debería ser la versión en la que un nuevo desarrollador debería centrarse:

- iPhone - 4S, 5, 5C, 5S, 6, 6 Plus, 6S, 6S Plus
- iPod Touch - generación 5 y 6
- iPad - generación 2, 3 y 4, Air, Air 2, iPad Pro
- iPad Mini - generación 1, Mini 2, Mini 3, Mini 4
- Apple TV (tvOS, un subconjunto de iOS)
- Apple Watch (watchOS, un subconjunto de iOS)

Encontrarás una lista detallada de los dispositivos iOS, sus capacidades y las versiones de iOS compatibles en Wikipedia[6].

La Arquitectura

Como ocurre en la mayoría de los sistemas operativos, la arquitectura iOS se define por capas de tecnologías para permitir que tu aplicación se ejecute en un dispositivo sin comunicarse directamente a nivel de hardware. Estas tecnologías se pueden pensar en capas o interfaces que están empaquetadas como frameworks que el desarrollador puede importar en sus proyectos de iOS. El framework principal se llama Cocoa Touch.

5 developer.android.com/about/dashboards

6 en.wikipedia.org/wiki/List_of_iOS_devices

Cocoa Touch

Aunque OSX e iOS son diferentes sistemas operativos, comparten mucho en común en lo que se refiere a frameworks, herramientas de desarrollo y patrones de diseño.

Apple potenció y amplió el framework principal para el desarrollo de aplicaciones de OSX, Cocoa, y le añadió soporte para las características únicas de iOS como los gestos táctiles, bautizando el resultado como Cocoa Touch. En Cocoa Touch se incluyen frameworks para construir interfaces de usuario, acceder a sensores como el acelerómetro y llevar a cabo tareas de conexión a internet y gestión de datos.

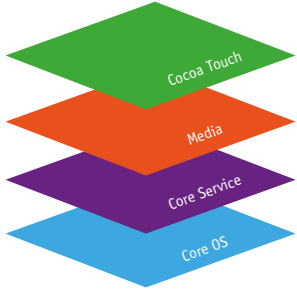


figura 1

Primeros Pasos en Programación iOS

Las aplicaciones móviles se clasifican comúnmente como "nativas", "web móviles" o "híbridas". En general, una aplicación iOS nativa se construye utilizando la plataforma de Apple, mientras que una aplicación híbrida utiliza una plataforma de terceros como Xamarin, Appcelerator o Phone Gap. Estas plataformas tratan de hacer posible el desarrollo para múltiples plataformas móviles utilizando un único lenguaje y conjunto de herramientas. Las aplicaciones web móviles suelen utilizar estándares HTML5 para crear algo que aparente ser una aplicación nativa pero ejecutable en el navegador web del dispositivo.

Junto con el SDK para desarrollar para iOS, Apple también ofrece un entorno de desarrollo integrado (IDE) llamado Xcode, para programar tanto para iOS como para OSX. A medida

que Xcode ha ido evolucionando, Apple se ha esforzado en proporcionar todas las herramientas necesarias para escribir, probar, monitorizar el rendimiento y enviar aplicaciones a la App Store, todo directamente desde Xcode.

Xcode

Apple lanzó Xcode en 2003 para escribir aplicaciones en OSX. La versión 3 de Xcode soportó el primer SDK para iPhone en 2008 y la versión más reciente es Xcode 7, lanzado con iOS 9 en septiembre del 2015. Xcode es un entorno de desarrollo integrado que se utiliza durante todo el proceso de desarrollo de aplicaciones. Interface Builder es una herramienta de diseño visual utilizada para diseñar y conectar vistas de la aplicación sin necesidad de escribir código y está integrado con Xcode. También se proporciona un simulador de iOS para permitir a los desarrolladores probar sus aplicaciones en todos los dispositivos actuales sin tener que instalar las aplicaciones en dispositivos físicos.

Interface Builder

Una discusión acalorada entre los desarrolladores iOS es si es mejor utilizar Interface Builder para diseñar visualmente la interfaz de usuario y el flujo de la aplicación o llevar a cabo todo de forma manual con código. En el pasado esto podía ser una preferencia personal, pero con nuevos dispositivos y tamaños de pantalla como el Apple Watch y el iPhone 6, Interface Builder resulta cada vez más necesario. Una de las principales diferencias entre iOS y Android era precisamente no tener que desarrollar para muchos tipos de dispositivos y tamaños de pantalla. Sin embargo, esta frontera es cada vez más difusa con iOS 9 soportando seis tamaños diferentes de pantalla. En lugar de soportarlos a todos por separado en tus aplicaciones, Interface Builder utiliza conceptos como Auto-Layout y Adaptive Layout para ayudar al desarrollador a

ser fácilmente compatible con todos los tamaños de pantalla. Interface Builder se ha visto mejorado con cada nueva versión de Xcode, por lo que es evidente que Apple prefiere que los desarrolladores lo aprovechen. Algo que un nuevo desarrollador iOS debe tener en cuenta.

Objective C

Objective C tiene sus orígenes en el sistema operativo NeXTSTEP, desarrollado en los ochenta, de donde se derivan OSX e iOS. Es un lenguaje de programación orientado a objetos que añade mensajería al lenguaje de programación C[7]. De hecho, C y C++ se pueden escribir junto a Objective C y algunos de los frameworks de iOS solamente proporcionan una API para C. Sin embargo, ha sido criticado por tener una sintaxis peculiar con una gran cantidad de asteriscos, arrobas y corchetes, que lleva a una curva de aprendizaje más pronunciada para desarrolladores procedentes de lenguajes más modernos como Java o C#, al mismo tiempo que proporciona una mejor legibilidad gracias a sus parámetros con nombre y nombres de clases y métodos largos. A lo largo del tiempo han habido mejoras incrementales, incluyendo el acceso a propiedades de objetos mediante sintaxis de punto, bloques, literales para colecciones y gestión de memoria a través de Automatic Reference Counting (ARC). Pero la necesidad de utilizar punteros y archivos de cabecera y el hecho de seguir acoplado a las limitaciones y riesgos del lenguaje C han hecho que Apple llegue a la conclusión de que es necesario un lenguaje más moderno.

7 http://en.wikipedia.org/wiki/Objective-C

Swift

En julio del 2010 Chris Lattner, Director y Arquitecto Sénior en el Departamento de Herramientas de Desarrollo de Apple, comenzó a implementar la estructura básica de un nuevo lenguaje de programación del cual solo unas pocas personas conocían. Se convirtió en un foco principal para el grupo de Apple Developer Tools en julio del 2013 y, casi un año después, Apple anunció en la World Wide Developer Conference (WWDC) un nuevo lenguaje de programación para iOS y OSX llamado Swift. Lattner afirma que Swift está influenciado por otros lenguajes como C#, Ruby, Haskell, Python y muchos otros[8].

La razón de que Apple considerara necesario crear Swift fue que la sintaxis de este lenguaje moderno es más concisa y fácil de aprender para los nuevos desarrolladores de iOS, incluyendo características nuevas como tipos de datos inferidos, declaraciones de estructuras de datos, tuplas, clausuras, punto y coma opcional y ausencia de punteros. Se ha sugerido que la apuesta de Apple por Swift es para asegurarse de que los desarrolladores de iOS permanecen interesados en herramientas de Apple y no estén tentados de usar otras plataformas que soportan lenguajes modernos para el desarrollo de iOS.

A principios de diciembre del 2015, Apple hizo Swift de código abierto[9] junto con un montón de herramientas relacionadas, frameworks y ejemplos. Apple está participando activamente con la comunidad en el desarrollo futuro del lenguaje a través de la petición de feedback, propuestas de nuevas características para el lenguaje y pull requests. Menos de una semana después de que el código fuera liberado, Swift ya ocupa la primera posición entre los lenguajes de

8 nondot.org/sabre

9 github.com/apple/swift

programación de código abierto en GitHub[10], superando a otros lenguajes populares como Ruby o PHP.

Herramientas de Rendimiento y Testeo

Además de proporcionar las herramientas para desarrollar aplicaciones iOS, Xcode también proporciona herramientas para monitorización y pruebas de rendimiento.

Instruments

Instruments permite a los desarrolladores recoger datos sobre el rendimiento y el comportamiento de sus aplicaciones iOS durante un período de tiempo. Algunas de las plantillas comunes disponibles permiten a los desarrolladores rastrear fugas de memoria, o detectar "puntos calientes" en la aplicación utilizando un rastreador. La herramienta Automation se utiliza para automatizar las pruebas de interfaz de usuario de tu aplicación iOS a través de scripts de prueba escritos por el desarrollador. Estos scripts se ejecutan fuera de la aplicación y simulan la interacción del usuario llamando a la API UI Automation. Se pueden ejecutar en un dispositivo o en el simulador.

XCTest Framework

XCTest es el framework de pruebas integrado con Xcode para proporcionar pruebas extensas de manera organizada y eficiente. Por defecto, los nuevos proyectos creados en Xcode utilizando una de las plantillas de aplicaciones añadirán un target de testeo para el proyecto. Esto permite al desarrollador escribir sus propias clases de pruebas unitarias, ejecutarlas y analizar los resultados utilizando el Test Navigator, todo desde dentro de Xcode.

10 github.com/showcases/programming-languages

Preparando el Entorno de Desarrollo

Después de crear una cuenta de desarrollador gratuita en developer.apple.com, recibirás autorización para descargar Xcode, código de ejemplo, vídeos y documentación. Los requisitos para instalar todas las herramientas de Xcode son un ordenador Mac con sistema operativo OS X 10.10 (Yosemite) y el SDK de iOS. Esta configuración permitirá la creación y prueba de aplicaciones iOS en el simulador de iOS. Para enviar aplicaciones a la App Store debes actualizar la cuenta de desarrollador con un pago de 99 dólares al año, lo que también te da acceso a las futuras versiones betas de Xcode e iOS.

Distribución

El método principal para el despliegue de aplicaciones es a través de la App Store. Cada aplicación presentada es revisada por el equipo de Apple para garantizar que cumple los requisitos y normas fijadas por la compañía. Esta es una diferencia importante respecto a la tienda de Google Play para aplicaciones de Android, ya que Google no revisa las aplicaciones sino que simplemente se asegura de que el código esté firmado.

Apple es muy estricta respecto a cómo las aplicaciones de terceros se ejecutan en iOS y utiliza la técnica de la caja de arena para garantizar la seguridad de las mismas y trata de evitar que cierto código malicioso o con errores pudiera comprometer el sistema operativo, otras aplicaciones o el propio dispositivo. Piensa en una caja de arena como una barrera virtual alrededor de la aplicación que define a qué recursos puede acceder la aplicación. Por ejemplo, una aplicación no tiene acceso a los directorios de archivos o recursos del sistema de otra aplicación que no sean accesibles a través de frameworks del SDK. Apple ha dado más control al usuario para conceder acceso a sus datos (por ejemplo, contactos, calen-

darios, fotos) y localización GPS. Los desarrolladores deben preparar las aplicaciones para los casos en que el usuario ha denegado su permiso.

Recursos de Aprendizaje

La popularidad del ecosistema de desarrolladores de Apple hace que existan una multitud de recursos de aprendizaje en diferentes formatos para ayudar a un nuevo desarrollador a empezar a programar para iOS, y muchos de ellos son gratuitos. Aprovechar este tipo de recursos supone que la curva de aprendizaje para dominar el desarrollo para iOS se reduzca considerablemente.

Websites y Blogs

— **Developer.Apple.com** contiene completas guías de referencia y de programación para aprender a desarrollar aplicaciones iOS, así como documentación de todas las clases de los frameworks públicos. La página web de la biblioteca está organizada por tipos de recursos, temas y frameworks, y además ofrece un buscador. Un documento importante que conviene leer antes de diseñar la primera aplicación que quieras enviar a la App Store son las iOS Human Interface Guidelines[11]. Este documento ofrece a los desarrolladores un conjunto de recomendaciones sobre las maneras aprobadas por Apple para diseñar aplicaciones y asegurar una experiencia de usuario positiva. La violación de estas recomendaciones supondrá, con toda probabilidad,

11 developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG

que la aplicación sea rechazada por la App Store durante su proceso de revisión previo a la publicación.

- **Swift.org**, el sitio oficial de la comunidad Swift.
- **RayWenderlich.com** se ha convertido en un sitio esencial de tutoriales iOS gratuitos escritos por su comunidad de desarrolladores con el objetivo de "tratar los temas más interesantes y complejos y exponerlos de manera sencilla para que todos puedan aprender (y que así todos podamos hacer aplicaciones increíbles)". El sitio se ha ampliado para ofrecer libros de programación y tutoriales en vídeo (con una suscripción de pago). Suscríbete a su podcast semanal para conocer las últimas noticias relevantes para desarrolladores y escuchar entrevistas con líderes de la comunidad de desarrolladores iOS.
- **iOS.devtools.me** es una página web creada por Adam Swinden que actualiza diariamente con las mejores herramientas para desarrolladores iOS y servicios back-end para ayudar en el desarrollo de aplicaciones. El contenido está organizado por categorías (diseño, gráficos, depuración...), por popularidad y por novedad. También proporciona un boletín semanal con los últimos contenidos del sitio.
- **iOSDevWeekly.com** es una revisión semanal de los mejores recursos en línea relacionados con el desarrollo para iOS. Dave Verwer gestiona el sitio y ofrece un boletín electrónico publicado todos los viernes.
- **Galloway.me.uk**, un blog de Matt Galloway, desarrollador iOS y autor afincado en Londres. Su libro "Effective Objective-C 2.0"[12] es muy recomendable cuando estés listo para comenzar a aprender características avanzadas y consejos sobre este lenguaje.

12 disponible vía http://www.amazon.com/Effective-Objective-C-2-0-Specific-Development/dp/0321917014

- **Merowing.info** es un blog del desarrollador, formador y conferenciante Krzysztof Zablocki, que ofrece tutoriales e información sobre el desarrollo en iOS a través de su experiencia como consultor. También es activo en la comunidad de código abierto, creando herramientas y librerías para desarrolladores iOS.
- **AshFurrow.com** es otro bloguero y desarrollador iOS popular, que plantea orgullosamente que el objetivo de su blog es "explorar las partes dolorosas de iOS". Es autor de varios libros de desarrollo en iOS y también un conferenciante activo involucrado en la comunidad de código abierto.
- **This Week in Swift** es un boletín semanal con las noticias más interesantes relacionadas con Swift, ejemplos de desarrollo, tutoriales y pequeñas cosas relacionadas con el desarrollo de iOS.

Formación con Vídeos Online

Como miembro del Programa para Desarrolladores de Apple, tienes acceso gratuito a todos los vídeos, código fuente y presentaciones de la World Wide Developer Conference, disponibles para descargar y ver en línea a través del sitio web o también desde la aplicación WWDC para iOS los últimos años. Por lo general, Apple publica los vídeos el día después de la presentación, a diferencia de unos años atrás, cuando había que esperar semanas para poder acceder después de la conferencia anual.

Lynda.com actualmente ofrece más de 30 cursos en vídeo vía subscripción sobre la iniciación al desarrollo iOS, incluyendo cursos para iOS 8. El código fuente de los proyectos está disponible para descargar en función del nivel de suscripción

elegido. También tienen una aplicación gratuita en la App Store para ver vídeos en dispositivos iOS.

Un recurso formativa gratuito popular en vídeo lo ofrece la iTunes University consistente en un curso de un semestre completo de iniciación al desarrollo para iOS, impartido por la Universidad de Stanford. Las clases se sumergen en profundidad en el lenguaje Objective-C y los frameworks de iOS. Los alumnos pueden incluso descargar los ejercicios de programación. Los vídeos se ven a través de iTunes o de la aplicación iTunesU en dispositivos iOS

Por último, YouTube tiene un buen número de vídeos gratis para el aprendizaje de desarrollo iOS, incluyendo un canal creado por Mohammad Azam[13] que indexa varios tutoriales para iOS.

Consideraciones Finales

Es un momento emocionante para ser parte de la comunidad de desarrolladores de iOS y esperamos que este capítulo sea de utilidad en la búsqueda de un punto de partida. Decir que las cosas cambian rápidamente es quedarse corto con todos los nuevos dispositivos, frameworks y servicios que se han lanzado en los últimos años. Pero no te dejes intimidar por la velocidad a la que la tecnología se mueve dentro del ecosistema de Apple. La mayor parte de los conceptos básicos en el desarrollo de una aplicación estándar se aplican ahora igual que como lo hacían en las primeras versiones de iOS. Por suerte, hay un sinnúmero de recursos disponibles para empezar y hacer crecer tus habilidades de desarrollo en iOS y la mayoría de ellos son gratuitos.

13 www.youtube.com/user/azamsharp

Cosas a considerar al crear tu primera aplicación iOS "Hola Mundo" y más allá:

- ¿Debería comenzar con Objective-C o con Swift?
- ¿Tiene más sentido utilizar Interface Builder para diseñar la interfaz de usuario o es mejor hacerlo en código?
- Aunque utilizar servicios de back-end como CloudKit facilita el desarrollo, ¿me estoy enclaustrando demasiado en la arquitectura de Apple? ¿Podría eso hacer que una versión en Android no pudiera acceder a ese mismo back-end?
- ¿Cuáles son las desventajas de desarrollar aplicaciones iOS fuera de Xcode (utilizando herramientas multiplataforma)? ¿Es la comunidad de usuarios lo suficientemente grande como para encontrar respuestas a las dudas? ¿Se mantienen sus productos al día con las últimas versiones de iOS?
- ¿Cómo es el entorno actual para ser un desarrollador independiente para iOS a tiempo completo?

Las respuestas a estas preguntas están más allá del alcance de este capítulo, pero los recursos expuestos pueden ayudarte a esquivar las dificultades en el desarrollo en iOS y ayudarte a avanzar más rápido en el camino de ser un desarrollador iOS experimentado. Buena suerte y bienvenido al club.



Windows

Gracias a la buena acogida del producto y al marketing agresivo de Microsoft, Windows 10 ha ganado más de 110 millones de usuarios de PC hasta noviembre del 2015. Con su "Universal Windows Platform" (UWP) puedes enfocarte a la más amplia gama de dispositivos:

- Familia de dispositivos móviles: teléfonos con Windows, phablets
- Familia de dispositivos de escritorio: tabletas, portátiles, ordenadores de sobremesa
- Familia de dispositivos de equipo: Surface Hub
- Familia de dispositivos IoT: dispositivos compactos como wearables o electrodomésticos
- Sala de estar: Xbox One
- Dispositivo de realidad aumentada: HoloLens

Tus principales puntos de entrada como desarrollador o proveedor de aplicaciones son *dev.windows.com* y *design.windows.com*.

El Ecosistema

Mientras que Windows sigue dominando el mercado del PC, la cuota de mercado de Windows 10 Mobile (anteriormente Windows Phone) sigue siendo pequeña. De acuerdo con el Kantar World Panel[1], la cuota de mercado oscila entre alrededor del 10% para algunos mercados europeos y Rusia y el 0,4% para Japón en el trimestre que terminó en octubre del 2015.

1 kantarworldpanel.com/global/smartphone-os-market-share

La marca Surface obtuvo su reconocimiento y al parecer se vendieron más tablets con Windows que iPads de Apple en octubre del 2015, según 1010data[2].

Gracias al buen rendimiento de Windows 10, Microsoft espera poder disminuir la brecha en aplicaciones, lo que a su vez hace que la oferta móvil de Windows 10 sea más atractiva. Hasta el momento esa estrategia parece funcionar, habiéndose dado el caso de varios actores importantes del sector de las aplicaciones que se han unido a la plataforma Windows últimamente. De acuerdo con el informe de Developer Economics del tercer trimestre del 2015, el 44% de los desarrolladores móviles están planeando adoptar Windows 10[3].

Lenguajes y Herramientas

Como desarrollador Windows puedes elegir una de las siguientes opciones de lenguaje y juego de herramientas para la interfaz de usuario:

- C# o Visual Basic con XAML
- JavaScript con HTML/CSS
- C++ con XAML
- C++ con DirectX

El IDE principal es Microsoft Visual Studio[4], que a día de hoy también soporta desarrollo para Android y multiplataforma/Cordova.

Gracias a las librerías de clases portables (PCL, del inglés Portable Class Libraries) y a los componentes del Windows Run-

2 winbeta.org/news/microsoft-beats-apple-online-tablet-sales

3 developereconomics.com

4 visualstudio.com

time puedes utilizar el lenguaje y el framework de interfaz de usuario que mejor encajen para cada módulo de tu aplicación.

Si quieres utilizar DirectX con C#, puedes usar Win2D[5] o librerías de juegos como MonoGame[6]. Entre la pila de la interfaz de usuario de XAML y DirectX ahora también puedes acceder a la capa intermedia con la API UI Composition[7], a la que también se puede acceder fácilmente desde los lenguajes .NET.

Aunque el escenario más común es el uso de XAML para aplicaciones y DirectX para juegos, también se pueden crear juegos con XAML y aplicaciones con DirectX, en función de las necesidades. También es posible alojar Direct3D dentro de una aplicación XAML. Esto se podría utilizar para mostrar un modelo 3D dentro de una aplicación XAML dirigida por eventos, o para crear fácilmente unos menús elegantes basados en Silverlight alrededor de un juego completo en DirectX.

Históricamente las aplicaciones de Windows Phone se solían crear con Silverlight, un modelo de aplicación que es muy similar al enfoque UWP XAML. Pese a que las aplicaciones basadas en UWP son el futuro, todavía es posible crear aplicaciones de Silverlight para dispositivos Windows 10 Mobile.

Por último, pero no menos importante, hay varias opciones para crear aplicaciones sin conocimientos profundos de desarrollo:

— **PowerApps**[8] se dirige a las empresas que quieran conectar varias fuentes de datos y crear aplicaciones para Android, iOS y Windows sin conocimientos de programación. Cuenta con el respaldo de los servicios Azure por debajo.

5 github.com/Microsoft/Win2D

6 monogame.net

7 msdn.microsoft.com/library/windows/apps/mt592880

8 powerapps.microsoft.com

- **Windows Phone App Studio**[9] te permite crear aplicaciones Windows mediante la ayuda de plantillas.
- **Project Siena**[10] proporciona un entorno WYSIWYG (del inglés What You See Is What You Get, "lo que ves es lo que obtienes") para crear aplicaciones Windows de negocios.

Microsoft Design Language

Microsoft lideró el paradigma de diseño "plano", que también influenció fuertemente a Android e iOS. Su característica más obvia es su interfaz única y fácil de usar, que se centra en la tipografía y el contenido. Este paradigma de interfaz de usuario llamado Metro, o Modern UI, o Microsoft Design Language[11] se ha extendido a la Xbox y Windows 8 y evolucionó en lo que se conoce como Microsoft Design Language 2 (abreviado como MDL2). Este paradigma de interfaz de usuario contiene los siguientes principios:

- **Content not Chrome**: elimina adornos innecesarios y permite que el contenido sea en sí mismo el foco principal de atención. También debes abstenerte de utilizar todos los píxeles disponibles, ya que los espacios en blanco dan equilibrio y énfasis al contenido.
- **Alive in motion**: añade profundidad a un diseño plano con animaciones ricas.
- **Typography is beautiful**: da protagonismo al uso de fuentes en Metro. La fuente Segoe de Windows, inspirada en Helvetica, encaja con el enfoque moderno de la interfaz.
- **Authentically digital**: el diseño no trata de imitar objetos

9 appstudio.windows.com

10 microsoft.com/projectsiena

11 wikipedia.org/wiki/Metro_(design_language)

del mundo real, sino que se centra en las interacciones disponibles para soluciones digitales.

Los diseñadores encontrarán inspiración e información en *design.windows.com*.

Un aspecto importante de la experiencia de usuario son las 'live tiles' (pequeños widgets que residen en la pantalla de inicio). Puedes actualizarlos programáticamente o incluso de manera remota mediante notificaciones push.

En tanto que las aplicaciones UWP se pueden ejecutar en una gran variedad de dispositivos, debes diseñar una interfaz de usuario adaptativa, de tal modo que se adapte bien a diferentes factores de forma y resoluciones. Tu rejilla de base para conseguir el mejor escalado debe ser de 4x4 píxeles efectivos.

Integrarse en la Plataforma

UWP proporciona un amplio conjunto de APIs que están disponibles en cada familia de dispositivos compatibles. Si deseas utilizar funcionalidad específica de una familia de dispositivos, puedes integrar su extensión del SDK y utilizar código adaptativo para llamar a funciones específicas de la familia. Normalmente, se utiliza `Windows.Foundation.Metadata.ApiInformation` para probar si una API específica de familia está presente:

```
bool isHardwareButtonsApiPresent =
if (isHardwareButtonsApiPresent)
{
}
```

MVVM

Para los desarrolladores de aplicaciones venidos de otras plataformas, el concepto de enlace de datos de XAML será una novedad. Para cada página debe haber un modelo de vista que incluya los datos para esa página. La vista en sí misma solo describe la interfaz de usuario, mientras que los datos que se muestran proceden del modelo de vista. Las clases del modelo contienen los datos en sí. Este concepto de un Modelo, una Vista y un Modelo de Vista (MVVM) facilita considerablemente el desarrollo de aplicaciones complejas.

Servicios

Utilizando la API Microsoft Graph[12] puedes acceder a una variedad de servicios endpoint de Microsoft.

También están disponibles notificaciones push[13] que pueden actualizar las 'live tiles' de tu aplicación. Además puedes utilizar en tu aplicación el espacio en la nube OneDrive[14] disponible de manera gratuita.

Continuum y Cortana

Continuum permite al usuario cambiar fácilmente entre utilizar ratón y teclado y utilizar el modo táctil en los dispositivos de la familia de escritorio. En los dispositivos móviles, Continuum incluso te permite cambiar a una pantalla remota con soporte de ratón y teclado para que puedas trabajar "como un jefe"[15] con tu teléfono. Los requisitos más importantes para soportar Continuum de la mejor manera pasan por diseñar tu aplicación de manera adaptativa y ponerla a prueba tanto utilizando un

12 graph.microsoft.io

13 msdn.microsoft.com/library/windows/apps/mt187203

14 dev.onedrive.com

15 microsoft.com/windows/continuum

ratón como en modo táctil. Para más detalles, consulta *blogs.windows.com/buildingapps/2015/12/07/optimizing-apps-for-continuum-for-phone*.

Es posible extender el "asistente personal" Cortana[16] mediante la definición de tus propios comandos de voz. Después puedes mostrar los resultados de la interacción tanto en el interior de tu aplicación como dentro de la propia Cortana.

Testeo y Analíticas

La automatización de pruebas está integrada en Visual Studio. Utiliza proyectos de pruebas unitarias para tus módulos y proyectos de pruebas codificadas de la interfaz de usuario para esta última.

La Windows Store proporciona varias estadísticas sobre analíticas como número de descargas, compras, salud de la aplicación y más[17].

Hay varias opciones para los desarrolladores que deseen recopilar datos y analíticas en tiempo de ejecución. Flurry[18] y Google Analytics[19] proporcionan herramientas de análisis y servicios compatibles con Windows. Los desarrolladores también pueden utilizar el servicio de Microsoft Application Insights[20]. También hay disponibles sólidas herramientas de monitorización de rendimiento en Visual Studio para controlarlo durante la fase de desarrollo.

16 msdn.microsoft.com/library/windows/apps/dn974233

17 msdn.microsoft.com/library/windows/apps/mt148522

18 flurry.com/flurry-analytics.html

19 googleanalyticssdk.codeplex.com

20 azure.microsoft.com/services/application-insights

Distribución y Monetización

Distribuye tus aplicaciones a través de la Windows Store, que soporta más de 200 países y regiones.

Aunque el contenido de la aplicación es revisado y restringido de una manera similar a la de la App Store de Apple, Microsoft proporciona directrices bastante claras para la solicitud de publicación, disponibles en el Dev Center[21]. Aunque las herramientas de desarrollo se proporcionan de forma gratuita, se requiere una cuenta de pago para instalar una aplicación en dispositivos a través de la Windows Store. Actualmente, una cuenta cuesta 19 dólares en un pago único (no anual) para desarrolladores individuales y 99 dólares para empresas. Esta cuenta también te permite publicar aplicaciones en las tiendas de Azure y de Microsoft Office. Las tarifas no se aplican a los estudiantes miembros del programa DreamSpark[22]. Puedes utilizar el Windows Certification Kit que viene integrado en Visual Studio para probar tu aplicación localmente antes de enviarla para publicación.

Las aplicaciones se gestionan por parte del cliente, no del dispositivo. Así que un usuario puede instalar tu aplicación en una variedad de dispositivos o familias de plataformas, como por ejemplo un PC de escritorio y un teléfono. Si creas una aplicación de tipo Universal Windows Platform puedes elegir si solo es necesario comprarla en una familia de dispositivos o si es necesario comprarla de nuevo en cada familia de dispositivos utilizada.

Aprende sobre opciones de monetización y promoción en *dev.windows.com/monetize* y *dev.windows.com/store-promotion*.

21 msdn.microsoft.com/library/windows/apps/dn764944

22 www.dreamspark.com

Aprender Más

Visita *dev.windows.com* para acceder a noticias, herramientas de desarrollo y foros.

El equipo de desarrollo publica artículos en su blog[23] y en su cuenta de Twitter *@wpdev*. Si quieres encontrar una gran colección de recursos de desarrolladores y diseñadores independientes, visita *reddit.com/r/wpdev* y *windowscentral.com/developers*.

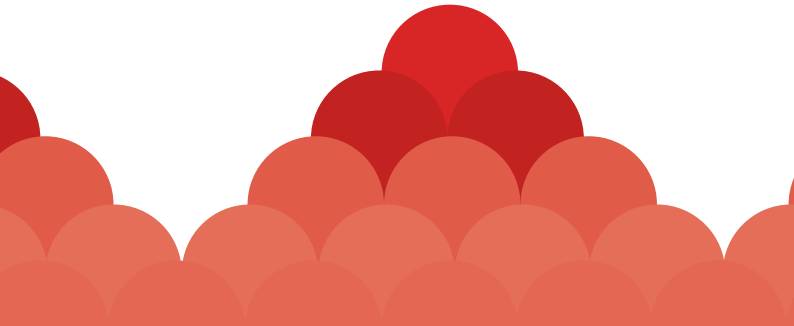
Puedes extender tus componentes, comportamientos y otras herramientas con varios kits de herramientas comerciales y de código abierto. Algunos ejemplos populares incluyen Telerik[24], Coding4Fun[25], Cimbalino[26], MVVM Light[27] y MvvmCross[28]. Para inspeccionar el árbol visual, los enlaces de datos y las propiedades de las interfaces de usuario XAML en tiempo de ejecución, tienes a tu disposición xaml spy[29].

Microsoft proporciona también una serie de vídeos en línea tanto para principiantes como para expertos y acerca de todo tipo de temas en su Virtual Academy[30] y en el sitio web de Channel9[31], incluyendo las grabaciones de todas las conferencias para desarrolladores de Microsoft.

23 blogs.windows.com/buildingapps
24 telerik.com
25 github.com/Coding4FunProjects
26 cimbalino.org
27 mvvmlight.codeplex.com
28 github.com/MvvmCross/MvvmCross
29 xamlspy.com
30 microsoftvirtualacademy.com
31 channel9.msdn.com

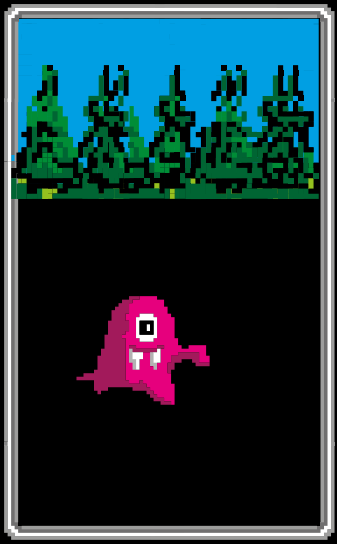
Puedes encontrar ejemplos de código en *dev.windows.com/samples* y en varios proyectos Codeplex[32]. Las guías "how-to" para desarrolladores de aplicaciones[33] suponen una buena visión general sobre planificación, diseño y desarrollo de aplicaciones Windows.

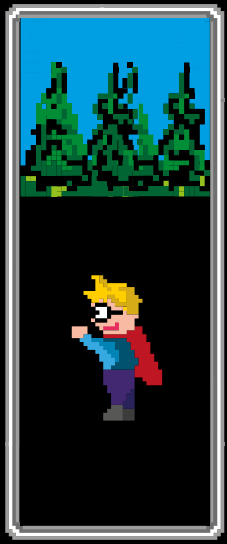
Si estás portando una aplicación existente desde iOS o Android, encontrarás ayuda en *msdn.microsoft.com/library/windows/apps/mt238321*.



32 codeplex.com

33 msdn.microsoft.com/library/windows/apps/xaml/mt244352





GUIDE
HP
30

FIGHT
MAGIC
DRINK
ITEM
RUN

Hacia Multiplataforma

Tantas plataformas y tan poco tiempo: ¿así que por qué no optar por multiplataforma? Hay plataformas más que suficientes para elegir: Android, iOS, Sailfish OS, Tizen, Ubuntu y Windows están esperando tus aplicaciones.

La mayoría de los promotores de aplicaciones, citando cierta famosa letra de una canción de Queen, le dirán al desarrollador: "Lo quiero todo, lo quiero todo, lo quiero todo... ¡y lo quiero ahora!". Así que la elección puede estar entre invertir en múltiples equipos paralelos de desarrollo o adoptar una estrategia multiplataforma.

Diferencias Clave Entre Plataformas Móviles

Si deseas hacer disponible tu aplicación a través de diferentes plataformas, tienes que superar algunos obstáculos. Ciertos retos son más fáciles de superar que otros:

Lenguaje de Programación

A estas alturas ya te habrás dado cuenta de que la mayoría de plataformas móviles publican sus propios SDKs que te permiten desarrollar aplicaciones en los lenguajes de programación soportados por las plataformas.

Sin embargo, estos lenguajes tienden a pertenecer a alguna de las pocas familias de lenguajes originales. La tabla siguiente ofrece una descripción general de estos y de las plataformas que los soportan:

Lenguaje	Soportado nativamente	Soportado como opción
C, C++	Sailfish OS, Ubuntu, Windows	Android (parcialmente, utilizando el NDK), iOS (parcialmente)
C# y Visual Basic	Windows	ninguno
Java	Android	ninguno
JavaScript	Tizen, Ubuntu, Windows	ninguno
Objective-C	iOS	ninguno
Swift	iOS	ninguno

1 por la plataforma, por ejemplo el lenguaje principal o el único para crear aplicaciones

2 por la plataforma, por ejemplo puede utilizarse como alternativa al lenguaje nativo pero por lo general no proporcionará el mismo nivel de acceso a las características de la plataforma.

Los frameworks multiplataforma pueden solventar las barreras del lenguaje de diferentes maneras:

- **Tecnologías Web**: este enfoque se basa en que la mayoría de las plataformas ofrecen soporte directo a las tecnologías web a través de 'webviews' integradas en aplicaciones nativas. Junto con HTML y CSS, esta aproximación también admite JavaScript.
- **Interpretación / Máquina Virtual**: el framework ofrece un motor para cada plataforma que interpreta un lenguaje común o específico del framework. Por ejemplo, una opción popular para el desarrollo de juegos es Lua scripting.
- **Compilación Cruzada**: es el santo grial de los frameworks multiplataforma, pero también se trata de la solución técnica más compleja. Permite escribir una aplicación en un lenguaje

y transcribirla al lenguaje nativo de cada plataforma, ofreciendo velocidad de ejecución nativa.

La mayoría de los frameworks también proporcionan un conjunto de APIs multiplataforma que te permiten acceder de un modo común a determinadas funciones del dispositivo o plataforma, como por ejemplo a las capacidades de geolocalización. Para funciones como mensajería SMS también puedes usar APIs de red que son independientes del dispositivo.

Versiones de Sistema Operativo

Las plataformas evolucionan y, tarde o temprano, ofrecerán características específicas de una versión que desearás aprovechar. Esto añade otra capa de complejidad a tu aplicación y también un desafío para las herramientas multiplataforma: a veces se quedan atrás cuando se publica una nueva versión del sistema operativo.

Interfaz de Usuario y Experiencia de Usuario

Un obstáculo difícil para el enfoque multiplataforma se debe a los diferentes patrones de interfaz de usuario (UI) y experiencia de usuario (UX) que prevalecen en las plataformas individuales.

Hoy en día la mayoría de las plataformas utilizan una variación del diseño plano promovido por primera vez por Windows Phone en 2007. Sin embargo, hay muchas diferencias y sutilezas entre las plataformas, por lo que transferir exactamente el mismo aspecto y percepción resultaría en una aplicación con apariencia torpe en otras plataformas.

Otro desafío clave con una interfaz de usuario multiplataforma uniforme es que puede comportarse de manera diferente a lo que están acostumbrados los usuarios de la interfaz de usuario nativa, dando como resultado que tu aplicación no "funcionará" para los usuarios. Un caso simple es no dar

soporte a una tecla de hardware correctamente, como por ejemplo la tecla de retroceso en una determinada plataforma. Otro reto es el valle de las sombras resultante de intentar imitar elementos nativos de la interfaz, que se parecen a los originales pero no funcionan de la misma manera. En lugar de imitar controles nativos, deberías usar unos de aspecto no nativo o, simplemente, limitarte a usar los "auténticos" y tirar hacia lo nativo.

Al dirigirte a los consumidores finales directamente (B2C), a menudo tienes que tener mucho más en cuenta la experiencia de usuario específica de plataforma que en aquellos otros casos en los que te diriges a usuarios de negocio (B2B). En cualquier caso, debes tener en cuenta que personalizar y adaptar la interfaz y la experiencia de usuario para cada plataforma puede suponer una gran parte de tu esfuerzo de desarrollo y es sin duda el aspecto más desafiante de una estrategia multiplataforma.

Soporte a la Integración en Pantalla de Inicio

La integración de tu aplicación en las pantallas de inicio de los dispositivos varía mucho entre las plataformas. En iOS únicamente puedes añadir una etiqueta con un número al icono de tu aplicación, en Windows puedes crear live tiles que añaden información estructurada, mientras que en Android es posible añadir un widget completo que muestre cualquier tipo de datos y emplee imágenes.

Utilizar la integración de tu aplicación en este sentido podría mejorar drásticamente la interacción con tus usuarios.

Soporte Multitarea

La multitarea permite que servicios en segundo plano y múltiples aplicaciones funcionen al mismo tiempo. Es otra de las características que se llevan a cabo de manera diferente

según el sistema operativo. En Android, Ubuntu y Sailfish OS hay servicios en segundo plano y puedes ejecutar varias aplicaciones al mismo tiempo, pero en Android el usuario no puede terminar las aplicaciones a voluntad, ya que esto lo realiza el sistema operativo automáticamente cuando los recursos escasean. En iOS y Windows disponemos de una selección limitada de tareas en segundo plano que pueden seguir funcionando después de salir de la aplicación. Así que, si los servicios en segundo plano pueden mejorar la operativa de tu aplicación, debes evaluar las estrategias multiplataforma cuidadosamente para asegurarte de que permiten el acceso más completo a las capacidades del teléfono en este sentido.

Consumo de Batería y Rendimiento

En estrecha relación con la multitarea está el consumo de batería que hace tu aplicación.

Mientras que la potencia de procesador se duplica aproximadamente cada dos años (la Ley de Moore dice que el número de transistores se duplica cada 18 meses), la capacidad de la batería, en cambio, se duplica solo una vez cada siete años. Esta es la razón por la que los smartphones pasan tanto tiempo conectados al cargador. Cuanto más cerca estás de la plataforma en una capa de abstracción multiplataforma, mejor puedes controlar el consumo de la batería y el rendimiento de tu aplicación. Como regla general, cuanto más tiempo necesita tu aplicación para arrancar y funcionar, menor abstracción puedes conseguir.

Asimismo, algunas plataformas tienen una gran variedad de resultados en rendimiento, sobre todo Android, cuya gama de dispositivos va desde lo dolorosamente lento hasta lo súper rápido.

Servicios Push

Los servicios push son una estupenda manera de aparentar que tu aplicación está activa incluso cuando no está en funcionamiento. En una aplicación de chat puedes, por ejemplo, enviar mensajes de chat entrantes al usuario mediante un mecanismo push. Como en el caso anterior, la manera en que los servicios push funcionan y los protocolos que utilizan se emplean de manera diferente y utilizan distintos tamaños de datos en cada plataforma.

Compra desde Aplicación

La opción de monetización más importante a día de hoy son las compras desde la aplicación. No hace falta decir que esto funciona de forma diferente en distintas plataformas. Para ver más detalles, consulta los capítulos dedicados a cada plataforma y el dedicado a la monetización.



Publicidad en Aplicación

Existen diferentes opciones para la visualización de anuncios en aplicaciones móviles. Algunas de ellas son soluciones de terceros independientes del fabricante. Sin embargo, los servicios de publicidad específicos de cada plataforma pueden ofrecer mayores ingresos y una mejor experiencia de usuario. Una vez más, estos servicios funcionan de manera diferente según la plataforma. El capítulo de monetización en esta guía proporciona más información sobre este tema.

Estrategias Multiplataforma

En esta sección se describen algunas de las estrategias que puedes emplear para implementar tus aplicaciones en diferentes plataformas.

Soporte Directo

Puedes soportar varias plataformas disponiendo de un equipo especializado para cada plataforma objetivo. Si bien esto puede consumir muchos recursos, es muy probable que te dé la mejor integración y experiencia de usuario en cada sistema. Una vía fácil de inicio es comenzar con una plataforma y luego progresar hacia plataformas adicionales una vez que tu aplicación se ha probado en el mundo real.

Las librerías de componentes pueden ayudarte a acelerar el desarrollo nativo, hay muchos de estos componentes tanto de pago como de código abierto disponibles para todas las plataformas.

Compartición de Recursos

Cuando mantienes varios equipos para diferentes plataformas puedes ahorrar mucho esfuerzo si compartes algunas estructuras de la aplicación:

- **Concepto y recursos**: esto se hace habitualmente de forma automática y consiste en compartir las ideas y los conceptos de la aplicación, el flujo de la interfaz de usuario, los mecanismos de entrada y salida y los recursos gráficos de la aplicación (pero debes ser consciente de la necesidad de dar soporte a las estructuras específicas de interfaz de usuario de cada plataforma).
- **Estructuras de datos y algoritmos**: ve un paso más allá compartiendo estructuras de datos y algoritmos entre plataformas.
- **Compartición de código del modelo de negocio**: utilizando compiladores de multiplataforma también puedes compartir el modelo de negocio entre las plataformas. Alternativamente, puedes utilizar un intérprete o una máquina virtual y un lenguaje común en un grupo de plataformas.
- **Abstracción completa**: algunas herramientas multiplataforma te permiten abstraer completamente el modelo de negocio, las vistas y el control de tu aplicación para diferentes plataformas.

Reproductores y Máquinas Virtuales

Los reproductores suelen proporcionar un conjunto común de APIs sobre distintas plataformas. Ejemplos famosos incluyen Xamarin[1] y Lua[2]. Este enfoque hace que el desarrollo sea muy fácil. Sin embargo, te vuelves dependiente del proveedor de la plataforma para incorporar nuevas características, y también tienes un reto cuando las funciones están disponibles en una única plataforma. A menudo los reproductores tienden a utilizar

1 http://xamarin.com xamarin.com]

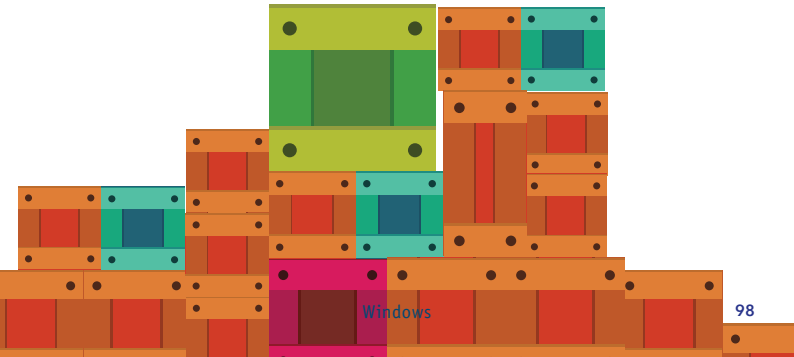
2 lua.org

una aproximación de "mínimo común denominador" respecto a las funciones ofertadas para mantener así la uniformidad entre las implementaciones para varias plataformas.

Compilación Cruzada

La compilación cruzada permite la programación en un lenguaje que después es transformado en el lenguaje específico de una plataforma diferente. En términos de rendimiento esta es a menudo la mejor solución de multiplataforma, aunque puede haber diferencias en comparación con las aplicaciones nativas. Esto puede darse, por ejemplo, cuando ciertas construcciones de programación no pueden ser traducidas de manera óptima desde el código fuente al lenguaje de destino.

Hay tres enfoques habituales para la compilación cruzada: traducción directa del código fuente, de forma indirecta mediante la traducción del código fuente en un lenguaje intermedio abstracto, y compilación directa en el formato binario de una plataforma. El enfoque indirecto suele producir código menos legible. Este es un problema potencial cuando se desea continuar con el desarrollo en la plataforma de destino usando el código fuente traducido como punto de partida.



Aplicaciones Web (Híbridas)

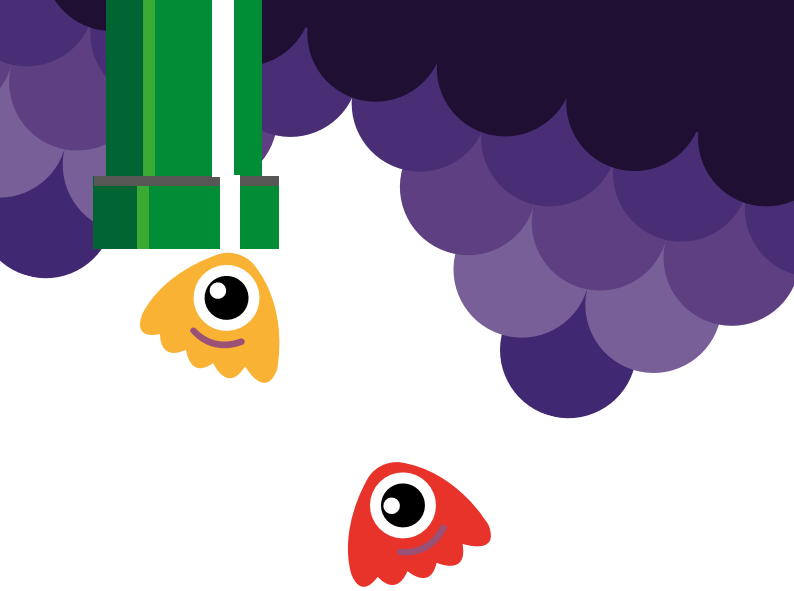
El desarrollo web híbrido consiste en embeber una 'webview' dentro de una aplicación nativa. El estándar para aplicaciones híbridas es la herramienta de código abierto Apache Cordova[3] (anteriormente conocida como PhoneGap).

Este enfoque permite acceder a funcionalidad nativa desde el interior de las partes web de tus aplicaciones y, al mismo tiempo, puedes utilizar código nativo para partes de tu aplicación donde el rendimiento o la experiencia del usuario sean aspectos críticos. Las aplicaciones híbridas te permiten reutilizar las piezas de desarrollo web a través de las plataformas elegidas. Lee el capítulo sobre tecnologías web para aprender más acerca del desarrollo web móvil.

ANSI C

Mientras que el HTML y la programación web se crean a partir de un nivel de abstracción muy elevado, también puedes elegir el camino opuesto utilizando ANSI C. Puedes ejecutar código ANSI C en todas las plataformas importantes, como Android, iOS y Windows. El principal problema con esta opción es que no puedes acceder a las APIs específicas de plataforma ni a los controles de interfaz de usuario desde ANSI C. Utilizar C es especialmente adecuado para algoritmos complejos como codificadores de audio. Las librerías correspondientes se pueden utilizar entonces en cada proyecto de aplicación para cada plataforma.

3 https://cordova.apache.org cordova.apache.org]



Encontrar el Framework Multiplataforma Adecuado

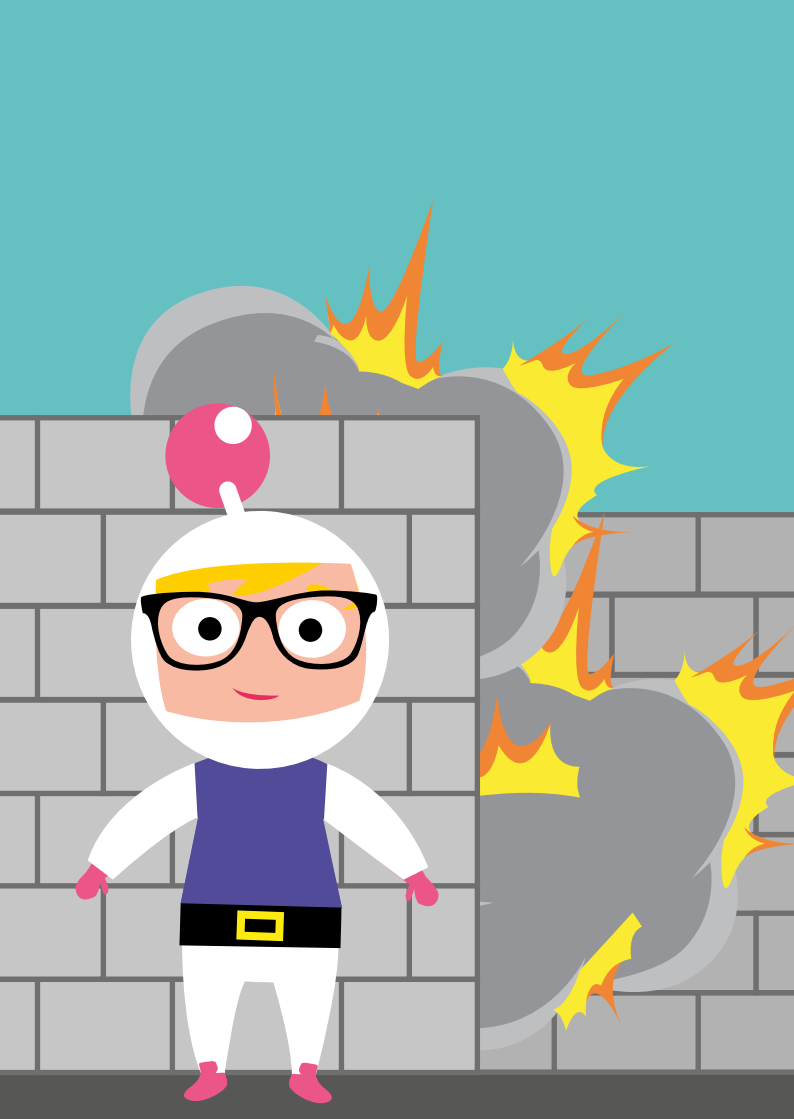
Existen tantas soluciones multiplataforma disponibles que hemos decidido no enumerarlas más en este libro. Si estás interesado, revisa por favor ediciones previas de esta guía[4]. Esperamos resucitar nuestra lista en formato digital en el futuro. Para una comparativa de los frameworks disponibles, puedes consultar el informe de research2guidance disponible en *research2guidance.com/cross-platform-tool-benchmarking-2014*.

A continuación te presentamos algunas preguntas que

4 obtenlas en

deberías plantearte al evaluar herramientas multiplataforma. No todas ellas serán relevantes para ti, así que sopesa las opciones convenientemente. En primer lugar ten una visión detallada de tu idea de aplicación, el contenido, su público objetivo y las plataformas de destino. Ten también en cuenta el grado de competencia en las distintas plataformas, tu presupuesto para marketing y el know-how de tu equipo de desarrollo.

- ¿Cómo funciona tu herramienta multiplataforma? ¿Qué lenguajes de programación y APIs puedo utilizar?
- ¿Puedo acceder a funciones específicas de la plataforma? En caso afirmativo, ¿cómo?
- ¿Puedo usar componentes nativos para la interfaz de usuario? En caso afirmativo, ¿cómo?
- ¿Puedo utilizar una construcción específica de la plataforma como base de mi desarrollo? ¿Qué pinta tiene el código fuente traducido/generado?
- ¿Es posible la integración con la pantalla de inicio?
- ¿Puedo controlar la multitarea? ¿Hay servicios en segundo plano?
- ¿Cómo funciona la solución con servicios push?
- ¿Cómo puedo utilizar las compra desde aplicación y la publicidad?
- ¿Se mantiene actualizado el framework a medida que el sistema operativo saca nuevas versiones?
- ¿Cuál es el rendimiento de la solución?



BY Daniel Kranz

Sitios para Móviles y Tecnologías Web

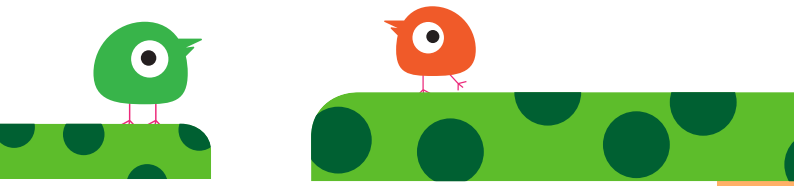
El desarrollo continuo de tecnologías web unido al incremento de dispositivos conectados a internet hace presagiar un gran futuro para aquellos deseosos de un incremento en la audiencia de los sitios webs móviles. El tráfico global de la internet móvil está creciendo rápidamente y ya ha superado el 52% en 2015[1]. El tiempo invertido en navegar por internet según dispositivo (móvil, tablet, ordenadores de sobremesa y televisor inteligente) varía mucho en diferentes puntos del planeta. La mayoría de regiones en donde el tráfico en internet vía móvil ya ha superado a los ordenadores de sobremesa son mercados en desarrollo o emergentes. Los mercados desarrollados como los Estados Unidos con una alta penetración de teléfonos inteligentes van a continuación. Cabe señalar que, si bien en los mercados de teléfonos de gama media se pasa un tiempo considerable realizando búsquedas y navegando sitios web con el móvil, los mercados con una alta penetración de smartphones muestran el nivel de vinculación más grande con aplicaciones móviles. Desde mayo del 2015, las aplicaciones móviles representan más de la mitad de todo el tiempo digital en los Estados Unidos[2].

El uso más obvio de las tecnologías web es para construir sitios web móviles, y este es el tema principal de este capítulo. Sin embargo, vale la pena destacar que las tecnologías web también se usan de manera intensiva en el desarrollo de aplicaciones móviles web e híbridas, soluciones multiplataforma e in-

1 statista.com/topics/779/mobile-internet

2 comscore.com

cluso aplicaciones nativas. Una gran ventaja de las tecnologías web es que ofrecen la ruta más fácil hacia el desarrollo móvil. Tecnologías web como HTML, CSS y JavaScript se han desarrollado plenamente a lo largo de muchos años. Además, son sin duda más fáciles de aprender que algunos de los complejos lenguajes necesarios para el desarrollo de aplicaciones nativas. Los sitios web para móviles y las aplicaciones web hacen los contenidos accesibles en casi cualquier plataforma con un menor esfuerzo en comparación con el desarrollo nativo para varias plataformas. Esto significa que los sitios web para



móviles tienen un alcance más amplio de manera automática. Por consiguiente, el desarrollo web móvil no solo ahorra tiempo y costes de desarrollo, sino que además proporciona una alternativa ajustada en costes y tiempo también en lo que se refiere a mantenimiento. Y ser independiente de las tiendas de aplicaciones te permite ofrecer cualquier contenido que quieras rápidamente y sin necesidad de suscribir las políticas de un tienda de aplicaciones.

No obstante, también hay puntos en contra. Las tecnologías web se esfuerzan por igualar el nivel de integración en la plataforma y el acceso directo a funciones de hardware que ofrece el desarrollo de aplicaciones nativas. Y además el rendimiento de las tecnologías web es altamente dependiente de la conectividad, sitios grandes como Facebook y LinkedIn tienen dificultades para lidiar con las limitaciones de memoria

y hay un déficit de herramientas de desarrollo en comparación a las disponibles para el desarrollo de aplicaciones nativas.

La monetización de sitios móviles también puede resultar difícil, ya que los usuarios esperan tener acceso a los sitios web de forma gratuita. La herramienta más común para su monetización es la integración de publicidad. Las soluciones de pago para sitios móviles se encuentran todavía en sus primeras etapas y tienden a ser bastante complejas de implementar. Los actuales instrumentos de monetización de las tiendas de aplicaciones ofrecen, por el contrario, una fácil instalación y un alto nivel de seguridad para el usuario final.

Si la monetización es uno de los requisitos clave, una estrategia híbrida o de aplicación web podría resultar una buena opción de compromiso. En este caso, el desafío clave es combinar las capacidades únicas de las tecnologías nativas y web para crear un producto verdaderamente amigable para los usuarios. En el capítulo sobre el tema multiplataforma de este libro aprenderás más sobre como crear este tipo de aplicaciones llamadas híbridas.

Como principio básico, los usuarios no deberían sufrir la frustración y la decepción de ser redirigidos a un sitio web que nunca termina de cargarse, exige una elevada transmisión de datos o simplemente no funciona. Por el contrario, en el peor escenario posible el usuario debería ser redirigido a un sitio básico pero que provea todo el contenido relevante. Antes de comenzar a desarrollar hay que tener en cuenta aspectos clave como las capacidades de los dispositivos que utiliza tu audiencia, sus hábitos de navegación y sus planes de datos.

Desde la perspectiva de la experiencia de usuario, Google ofrece las 10 mejores prácticas para que pequeñas y medianas empresas consigan conversiones[3]:

3 www.dudamobile.com/webinar/Google_DudaMobile_Webinar.pdf

- **Piensa en los pulgares**: diseña tu sitio de manera que incluso unas manos grandes puedan interactuar con él.
- **Diseña para visibilidad**: asegúrate de que tu contenido puede ser leído si el usuario extiende su brazo completamente.
- **Simplifica la navegación**: una navegación clara, jerárquica y con scrolling vertical facilitan el acceso a la información.
- **Hazlo accesible**: idealmente, tu sitio móvil debería funcionar en cualquier dispositivo y en ambas orientaciones, vertical y horizontal.
- **Hazlo fácilmente convertible**: destaca la información que te ayudará a conseguir conversiones.
- **Hazlo local**: incluye funcionalidades que ayuden a la gente a encontrarte y contactarte.
- **Redirige a sitios web móviles**: da a los usuarios la oportunidad de utilizar la versión web clásica de escritorio, pero también facilita volver al sitio móvil.
- **Mantenlo ágil**: ayuda a los usuarios de móviles, diseña tu sitio para que cargue rápido y haz que sea fácil hacer una lectura superficial de la información.
- **Hazlo cómodo**: ofrece el mayor número posible de funcionalidades de la versión completa del sitio web en la versión móvil.
- **Aprende, escucha e itera**: los buenos sitios móviles están centrados en el usuario; es decir, se han creado contando con el feedback de la audiencia.

Google también ha implementado cambios en su modelo de búsquedas móviles y ha anunciado que penalizará a los sitios que no sigan estas recomendaciones. Echa un vistazo al

sitio para desarrolladores de Google[4] para estar al día de las novedades respecto a cómo optimizar tu sitio móvil.

HTML5

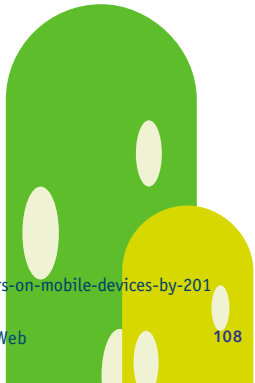
HTML5 es uno de los factores clave que hacen que los desarrolladores sopesen el desarrollo de sitios móviles y aplicaciones web en lugar de aplicaciones nativas. Aspectos como un look-and-feel similar al de aplicaciones nativas combinado con un único código base para una serie de dispositivos populares, la capacidad de acceder a hardware del dispositivo como la cámara y el micrófono, el almacenamiento de datos en los dispositivos para poder utilizarlos sin conexión y la optimización de la página web según el tamaño de pantalla hacen de HTML5 una atractiva alternativa para el desarrollo de aplicaciones nativas.

Sin embargo, HTML5 depende de un soporte universal por parte de los navegadores y es exactamente esta parte la que escasea actualmente. El ex-CTO de Facebook Brent Taylor en una ocasión describió la situación de la siguiente manera: "Hay una desenfrenada fragmentación de la tecnología en los navegadores móviles, por lo que los desarrolladores no saben qué parte de HTML5 pueden utilizar. HTML5 es promovido como un estándar único, pero viene en diferentes versiones para cada dispositivo móvil. Cuestiones como la aceleración por hardware y la gestión de derechos digitales están implementadas de manera inconsistente. Eso hace que sea difícil para los desarrolladores escribir software que funcione en muchas plataformas diferentes y llegue a un público amplio". Desafortunadamente esto todavía es cierto, aunque la situación mejora día a día.

4 developers.google.com/webmasters/smartphone-sites

Para más información acerca de la compatibilidad con navegadores, visita el sitio HTML5Test[5]. En él encontrarás tanto una introducción como un análisis profundo de las características de HTML5 según su soporte por los diferentes navegadores web. Facebook también ha desarrollado ringmark[6], que pone a prueba los navegadores web en tres niveles de soporte de características de HTML5, lo cual ayuda a los desarrolladores a comprobar rápidamente el nivel de soporte de múltiples navegadores web móviles (y de escritorio).

En resumen, casi todo el mundo en el negocio móvil está de acuerdo en que HTML5 tendrá éxito a largo plazo y especialmente en los dos últimos años se ha visto una rápida adopción del mismo. ABI Research estima que los dispositivos móviles compatibles con HTML5 existentes en el mundo serán 2.100 millones a finales del 2016[7]. Los sistemas operativos incrementarán gradualmente el soporte a características y navegadores de HTML5 para aumentar la velocidad de su adopción. Plataformas de código abierto como Sailfish, Tizen y Ubuntu también deberían favorecerla.



5 html5test.com/results/mobile.html

6 rng.io

7 www.abiresearch.com/press/21-billion-html5-browsers-on-mobile-devices-by-201

La Fragmentación Requiere Adaptación

El mayor reto del desarrollo web móvil es la fragmentación. En teoría, todos los dispositivos con conexión a internet pueden acceder a cualquier sitio móvil a través de un navegador. La realidad, sin embargo, es que los desarrolladores necesitan atender al creciente número de navegadores y dispositivos con diferentes niveles de capacidades de software y hardware para adaptar y optimizar el contenido web móvil.

En términos generales, existen dos enfoques para optimizar el contenido para dispositivos móviles: adaptación del lado del cliente o del lado del servidor.

- La adaptación del lado del cliente hace uso de una combinación de CSS y JavaScript ejecutándose en el dispositivo para ofrecer una experiencia idónea para móviles.
- La adaptación del lado del servidor utiliza el servidor para que ejecute lógica antes de que se suministre el contenido al cliente.

La siguiente sección ofrece una revisión de técnicas tanto del lado cliente como del lado servidor para hacer los sitios webs accesibles a la mayoría de dispositivos tanto actuales como futuros con acceso a internet.

Adaptación del Lado Cliente

Diseño Web Adaptativo (Responsive)

En su formato más simple, el diseño adaptativo se compone de una rejilla flexible, imágenes flexibles y CSS media queries para adaptarse a una serie de resoluciones de pantalla o tipos de dispositivos.

Esto por sí solo proporciona una experiencia adaptada al

dispositivo en un rango limitado de ellos y no puede adaptarse a contenido sofisticado. El mismo contenido se sirve a todos los dispositivos.

Pros

- Una adaptación completa en el lado cliente asegura que no haya impacto en la infraestructura existente.
- Posibilita un ajuste automático de contenido y diseño.

Contras

- El mismo contenido disponible en el sitio web estará disponible también en la versión móvil (visible o no).
- El peso del sitio tendrá un impacto significativo en términos de rendimiento en los dispositivos móviles.
- Es una aproximación general más que una optimización real a un entorno móvil.

Unido a componentes del lado del servidor y conocidos habitualmente como RESS (Responsive Design + Server Side Components), el diseño adaptativo también se puede utilizar para generar sitios web móviles y de escritorio más complejos.

Mejoras Progresivas

Las mejoras progresivas tienen la capacidad de responder a toda la gama de dispositivos móviles. Se envía una sola página HTML a todos los dispositivos. A partir de ahí, se utiliza código JavaScript para construir progresivamente la funcionalidad adaptada a un nivel óptimo para el dispositivo en particular. Como solución exclusivamente móvil, el principal inconveniente es el rendimiento. La construcción progresiva conlleva tiempo de ejecución y varía de acuerdo con el dispositivo y la red. Como solución tanto para escritorio como móvil, su

principal inconveniente es que un único documento HTML se envía a todos los dispositivos. Un framework muy conocido que hace uso de la mejora progresiva es jQuery Mobile[8].

Pros

- Una adaptación completa en el lado cliente asegura que no haya impacto en la infraestructura existente.
- Posibilita un ajuste automático de contenido, funcionalidad y diseño.

Contras

- Pérdida de control, en tanto que la detección es gestionada por el navegador.
- La detección de navegador todavía dista mucho de ser perfecta.
- La detección realizada en el lado cliente tiene impacto sobre el rendimiento global del sitio.
- El mismo HTML es servido a todos los dispositivos.

Adaptación del Lado Servidor

Bases de Datos de Dispositivos

Las bases de datos de dispositivos detectan a cada dispositivo que accede a la página web y devuelven una lista de las capacidades del dispositivo al servidor. Esta información se utiliza entonces para servir un sitio móvil acorde con las capacidades del dispositivo. La adaptación del lado del servidor es una de las soluciones más antiguas y fiables. Algunas bases de

8 jquerymobile.com

datos populares de dispositivos son WURFL[9] y DeviceAtlas[10]. El principal inconveniente de las bases de datos de dispositivos es que la mayoría solo están disponibles bajo licencia comercial.

Pros

- La solución más habitual (Google, Facebook, Amazon y similares).
- Máximo control.
- Es posible la optimización según dispositivo (por ejemplo iPhone, Samsung Galaxy y similares).

Contras

- Los repositorios de descripción de dispositivos (en inglés Device Description Repositories) están orientados al hardware.
- Aparte de los datos, es necesario un mecanismo de detección (un simple 'User-Agent' no es suficiente).

Diseño Web Adaptativo + Componentes del Lado Servidor (RESS)

Realmente lo mejor de ambos mundos, la combinación de la adaptación en lado cliente y servidor asegura un alto rendimiento gracias a la adaptación del lado del servidor y garantiza que las capacidades obtenidas se pueden utilizar para enriquecer la experiencia móvil para las visitas posteriores. Esta aproximación se conoce como RESS (Responsive Web Design

9 wurfl.sourceforge.net

10 deviceatlas.com

with Server Side Components). Naturalmente es costosa y, por tanto, más común en organizaciones de gran tamaño.

Las soluciones RESS o de adaptación híbrida están disponibles comercialmente a través de compañías como Sevenval[11] y Netbiscuits[12], y como soluciones en la nube respaldadas por la comunidad, por ejemplo FITML[13].

Mejor Entrada de Datos

Con teclados pequeños y a menudo en pantalla, introducir texto puede resultar engorroso y lento, especialmente si el usuario tiene que introducir números, direcciones de correo electrónico o textos similares. Afortunadamente, los desarrolladores pueden especificar el tipo esperado de datos y los smartphones mostrarán entonces el teclado más apropiado en pantalla. mobileinputtypes.com proporciona varios ejemplos claros y concisos.

Mejor Rendimiento

Los usuarios de móviles esperan que los sitios web tarden en cargar no más de 2-5 segundos. Esto a día de hoy supone un reto, especialmente para los sitios móviles más complejos. Ten en cuenta que la localización y la red utilizadas pueden tener un efecto drástico en el rendimiento del sitio. Aunque hay factores que quedan fuera de tu control, las siguiente secciones ofrecen trucos para reducir el volumen de datos transferidos, el contenido y las peticiones HTTP para así minimizar el tiempo de carga y mejorar el rendimiento.

11 sevenval.com

12 netbiscuits.com

13 fitml.com

Reducir el Volumen de Datos Transferidos

Activa GZIP cuando sirvas un sitio. Cambia los tamaños de imágenes y ajusta su calidad de imagen según sea la calidad de la red.

Reducir el Contenido

Tanto la carga del sitio como la de los contenidos auxiliares son cada vez más importantes. Reducir en lo posible elementos como los archivos JavaScript y CSS ayuda a reducir el tiempo global de carga. Múltiples archivos del mismo tipo pueden ser comprimidos en uno y los espacios en blanco pueden ser eliminados. De este modo se acorta el código, pero se sigue comportando del mismo modo. Todo esto puede resultar en un menor número de llamadas y, finalmente, en un mejor tiempo de carga.

Al mismo tiempo, es importante que el usuario comprenda qué está ocurriendo. Si el contenido está en proceso de carga es importante que el usuario sea consciente de ello y no mostrarle una página o espacio en blanco. Una experiencia sin contratiempos es primordial para cualquier experiencia móvil, y esto incluye cualquier punto desde el sitio hasta la carga de los contenidos en el mismo y todas aquellas animaciones que los acompañen.

Reducir las Peticiones HTTP

Transforma imágenes, scripts y estilos y utiliza la caché de aplicaciones. Cuando sea posible, reduce el número de llamadas, el tamaño de los archivos y el contenido. El principal beneficio es que los scripts se transmiten en una sola llamada por página, el tráfico HTTP se minimiza y los scripts básicos se almacenan en la cache de la aplicación. La implementación no afecta a la recarga y los scripts pueden alojarse igualmente en la cache pública (CDN).

Para más detalles y recomendaciones sobre rendimiento de webs móviles estudia la presentación de Roland Guelle en Slideshare[14].

Aplicaciones Híbridas y Multiplataforma

¿Armado con conocimientos en HTML5, CSS y JavaScript y con ganas de darle un empujón al desarrollo de aplicaciones? El desarrollo de aplicaciones híbridas podría ser la mejor ruta para ti. Phone Gap de Adobe[15] es solo un ejemplo de los frameworks multiplataforma y herramientas disponibles para crear, distribuir y actualizar un conjunto de sitios y aplicaciones convenientemente controlados a través de un único sistema. Lee también el capítulo sobre multiplataforma en esta misma guía para aprender más sobre tus opciones en esta área.

Pros

- Utiliza los conocimientos que ya tienes.
- Una única base de código para múltiples plataformas (iOS, Android, Windows, etc.).

14 www.slideshare.net/sevenval/mobile-web-performance-dwx13

15 phonegap.com

— Ahorro de tiempo y dinero al desarrollar para múltiples plataformas.
— Distribución tanto a través de tiendas de aplicaciones como del navegador web móvil para llegar a una mayor audiencia.

Contras

— Cuenta con algunos problemas de rendimiento cuando tratas de emular funcionalidades nativas complejas.
— Imitar interfaces de usuario nativas supondrá añadir más tiempo y esfuerzo a tu proyecto.
— Riesgo de ser rechazado por las tiendas de aplicaciones (especialmente la de Apple) si tu aplicación no parece lo suficientemente nativa.
— Algunas características de dispositivos o sistemas operativos pueden no estar soportadas.

Testear Tecnologías Web

Se puede probar de qué modo funcionan las tecnologías web en varios teléfonos móviles de múltiples maneras. La más simple es probar el sitio o aplicación web en una variedad de navegadores en dispositivos móviles. Esto debería incluir una combinación de los navegadores móviles más populares, por ejemplo tomando como base los datos públicos disponibles en línea[16]. El conjunto de dispositivos se puede refinar mediante el análisis de datos de los registros web existentes y otras fuentes de datos similares. Al mismo tiempo, realizar pruebas

16 gs.statcounter.com



en varios factores de forma ayuda a descubrir problemas de diseño y formato.

En cuanto a las pruebas automatizadas, WebDriver[17] es el framework predominante. Existen dos enfoques complementarios:

1. Testeo automático utilizando controles WebView en Android e iOS.
2. Simulación del User-Agent (spoofing) utilizando Google Chrome o Mozilla Firefox configurados para emular a varios navegadores web móvil.

17 seleniumhq.org/projects/webdriver

Ambas aproximaciones tienen pros y contras:

- Las WebViews incrustadas se ejecutan en el sistema operativo de la plataforma de destino. Es probable que encuentren muchos errores de comportamiento. Sin embargo, la configuración es más enrevesada y los sistemas operativos de algunas plataformas no están soportados.
- Con la técnica del spoofing se puede engañar a los servidores web para que traten al navegador como si accediera mediante una gran variedad de navegadores, incluyendo navegadores móviles no disponibles con la WebView incrustada, por ejemplo el teléfono Nokia Asha 201. Sin embargo, ni el comportamiento resultante ni el renderizado son realistas, así que muchos errores permanecerán no detectados, mientras que aparecerán algunos falsos positivos que no se darán en los dispositivos reales.

Aprender Más

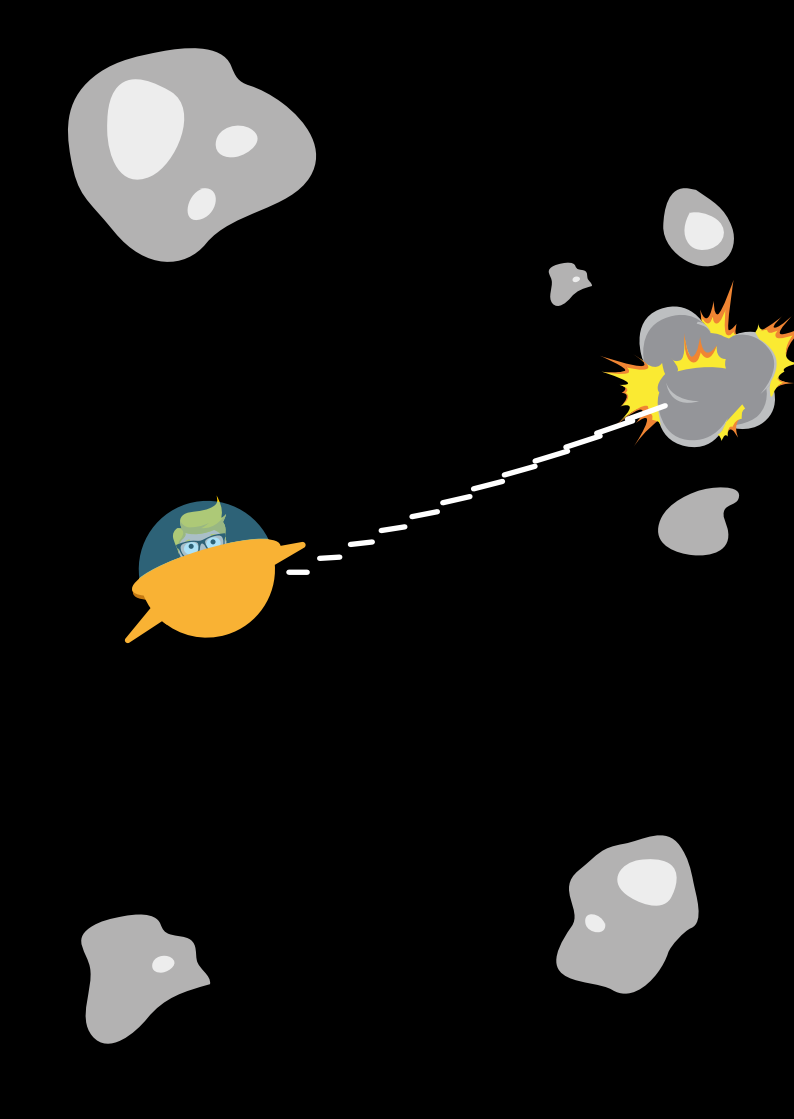
Online

- **W3Schools y CSS Tricks** (buenos recursos para iniciarse en HTML, CSS y JavaScript): *w3schools.com*, *css-tricks.com*
- **HTML5 Rocks** (excelente recurso sobre HTML5 que incluye tutoriales, presentaciones, artículos, etc.): *html5rocks.com/en*
- **Breaking the Mobile Web** (Max Firtman, el autor de múltiples libros acerca de programación web móvil, ofrece noticias actualizadas en su blog dedicado a móviles): *mobilexweb.com*
- **Mobi Thinking** (el recurso de DotMobi para vendedores,

con análisis y opiniones de expertos en marketing móvil): *mobithinking.com*
- **Testing (Mobile) Web Apps**: *docs.webplatform.org/wiki/tutorials/Testing_web_apps*
- **Investiga qué funcionalidades funcionan en todas las áreas web**: *caniuse.com* y *beta.theexpressiveweb.com*
- **WHATWG** (la página de la comunidad HTML): *whatwg.org*
- **Word Wide Web Consortium** (la organización que define los estándares web): *w3.org*

Libros

- **Mobile First** de Luke Wroblewski
- **Adaptive Web Design: Crafting Rich Experiences with Progessive Enhancement** de Aaron Gustafson y Jeffrey Zeldman
- **Responsive Web Design** de Ethan Marcotte
- **Programming the Mobile Web** de Max Firtman
- **jQuery Mobile: Up and Running** de Max Firtman



Aplicaciones Empresariales

Los responsables de decisiones corporativas ven actualmente las aplicaciones empresariales móviles como un factor estratégico, una necesidad, más que como una mera entrada en una hoja de cálculo de contabilidad. Las aplicaciones empresariales internas son capaces de reducir la latencia en la transferencia de información dentro de una organización, incrementando la agilidad del trabajador al hacer disponibles macrodatos y datos de carácter competitivo en cualquier momento y en cualquier lugar. Las aplicaciones también pueden permitir a las empresas estrechar lazos con sus clientes, proveedores y consumidores finales, etc. Ejemplos de aplicaciones empresariales incluyen software para puntos y personal de ventas, de respuesta a emergencias, de gestión de inventario o cadena de suministro y también de marketing B2C.

Puede parecer una obviedad, pero el riesgo más importante en este momento es **no** tener una estrategia móvil corporativa. Las empresas están tomando el enfoque **Móvil para Todos** en contraste a limitar su uso a la alta dirección, como ocurría en el pasado. Para conseguir esto, el enfoque tradicional del departamento de TIC de comprar y distribuir dispositivos en la estructura de gestión ya no es la única estrategia a utilizar. Hemos pasado del BYOD (Bring Your Own Device, en castellano Trae Tu Propio Dispositivo) al BYOx (Bring Your Own Anything, en castellano Trae Tu Propio...), lo que incluye aplicaciones, contenidos, herramientas y entornos de desarrollo, e incluso wearables, permitiendo al personal utilizar sus dispositivos personales para conectarse a la infraestructura TIC, descargar contenido seguro y usar aplicaciones empresariales. Con la llegada de BYOx, una empresa se expone a riesgos que tradicionalmente no se consideraban en la estrategia TIC corporativa. La

adopción temprana en la empresa de una estrategia móvil bien pensada y ejecutada es clave para asegurarse de que los datos están protegidos en todo momento.

Desde el punto de vista del desarrollador, el sector empresarial tiene mucho que ofrecer: en comparación con los desarrolladores de aplicaciones B2C tradicionales, aquellos que crean aplicaciones empresariales tienen el doble de probabilidades de estar ganando más de 5.000 dólares mensuales por aplicación y casi el triple de probabilidades de ganar más de 25.000, según el informe Developer Economics del tercer trimestre del 2014[1].

Puntos Clave de las Aplicaciones Móviles para Definir Iniciativas Empresariales

- Reducción de costes comparados con los sistemas existentes
- Racionalización de los procesos de negocio
- Ventaja competitiva con el acceso directo a datos actualizados de manera inmediata
- Incremento de la satisfacción y efectividad del empleado
- Rápida respuesta en comparación con procesos existentes
- Análisis y utilización de macrodatos

1 www.developereconomics.com/report/next-gold-rush-enterprise-apps

Estrategia Empresarial

Hoy en día, muchas empresas tienen un Chief Mobile Officer (CMoO) o han ampliado el alcance de sus CIOs. El trabajo del CMoO es coordinar las tendencias móviles y hacer de puente entre el negocio y las TIC. Dependiendo del tamaño y el enfoque de la empresa, su trabajo consiste también en construir un equipo interno de desarrollo de software móvil o coordinar la cooperación con una agencia de desarrollo externa. Para asegurarse de que el software móvil ofrece lo que los empleados/usuarios quieren, que sea técnicamente factible y que todo se ajuste a la estrategia global de la empresa, el director podría considerar la creación de un Consejo de Innovación Móvil (MIC, en inglés Mobile Innovation Council) o Centro de Excelencia (COE, en inglés Center of Excellence) que deben incluir a miembros clave como los representantes capacitados del equipo de desarrollo de aplicaciones móviles, las partes interesadas en el tema móvil en la empresa y, lo más importante, los usuarios finales de varios departamentos con experiencia en los procesos de negocio más relevantes.

Los temas en los que el CMoO/CIO debe centrarse en sintonía con el MIC/COE incluyen los siguientes:

- **Estrategia**: visión y dirección de la estrategia móvil global y las aplicaciones.
- **Políticas de gobernanza**: Bring Your Own Device (BYOD) versus Choose Your Own Device (CYOD), que básicamente es la diferencia entre una política de Gestión de Aplicaciones Móviles (MAM, del inglés Mobile Application Management) y una de Gestión de Dispositivos y Seguridad Móviles (MDM, del inglés Mobile Device Management & Security).
- **Especificaciones de las aplicaciones**.
- **Libro de ruta de las aplicaciones**.

- **Planificación presupuestaria**.
- **Visto bueno**: aprobación de las aplicaciones para producción.
- **Despliegue de aplicaciones**: feedback temprano de demos y prototipos, testeo y despliegue masivo.
- **Incentivos**: cómo incrementar la adopción y el uso de las aplicaciones creadas.

En cuanto a la adopción comercial, el desarrollo de aplicaciones corporativas es muy popular actualmente. Las empresas que programan aplicaciones corporativas para terceros o los mánagers de proyecto deseosos de adoptar una estrategia móvil interna corporativa solían preguntarse: "Todo esto suena muy bien, ¿pero por qué lo necesitamos?". Esto se ha convertido ahora en un "El móvil nos va a dar ventajas competitivas y hará más potente nuestra fuerza de trabajo", lo que supone una razón irresistible para que una empresa adopte una estrategia móvil.

Puntos Clave en los Casos de Negocio para Aplicaciones Móviles Empresariales

- Crear un plan de visión para nuevas aplicaciones móviles, en más dispositivos, y conocer cómo ayudarán, darán forma y harán más potente a tu empresa.
- Crear un manifiesto de definición de aplicación (ADS, del inglés Application Definition Statement) para cada aplicación, especificando propósito y audiencia objetivo.
- Crear un presupuesto para dispositivos y actualizaciones.
- Crear un plan para una estrategia de gestión de aplicaciones y dispositivos y para una infraestructura de seguridad.
- Crear un plan para el equipo de desarrollo de aplicaciones

móviles utilizando una plataforma de desarrollo fiable a futuro (como MADP, frameworks, etc.).

Hacer Móviles los Sistemas Existentes

Si ya estás ofreciendo un sistema a tus clientes pero aún no ha sido adaptado a móviles, tienes varias decisiones que tomar. Es crítico que comprendas en profundidad el impacto de añadir una oferta móvil a tu sistema antes de comenzar a implementar la solución elegida. Algunas de las razones más habituales para hacer móvil tu producto pueden ser por ejemplo utilizar las características de los teléfonos, como la cámara o el GPS, o simplemente ganar la capacidad de recabar información sobre la marcha, sin necesidad de conexión a internet. Debes asegurarte de que te adaptas al mundo móvil por las razones correctas, pues el consiguiente soporte, mantenimiento y desarrollo de una oferta móvil se convertirá en una nueva línea de producto respecto a tu modelo original e implicará algunos costes permanentes.



Puntos Clave al Decidir Hacer Móvil un Sistema Existente

- Define con claridad las razones para ello y asegúrate de que son suficientemente fuertes como para dar el paso hacia lo móvil.
- Comprende las diferencias entre móvil y sobremesa. No te limites a copiar el sistema preexistente. Por ejemplo, en vez de utilizar un formulario para capturar la información, podrías grabar audio y subirlo a tu sistema, permitiendo a los usuarios tomar notas rápidamente sin la necesidad de escribir en un dispositivo de tamaño reducido.
- No intentes implementar todas la funcionalidades de tu sistema existente; implementa solamente las importantes y hazlo de una manera que encaje con la filosofía móvil.
- Asegúrate de que conoces qué dispositivos utilizan tus clientes y qué funcionalidades son las que más interesa hacer móviles.
- Ten bien definida tu estrategia de pruebas en móviles, cubriendo pruebas en múltiples plataformas y en múltiples dispositivos y sistemas operativos.

Gestión de Dispositivos y Aplicaciones en la Empresa

Al desarrollar una aplicación empresarial, siempre hay que tener en cuenta que el hardware que contiene datos confidenciales de la empresa se puede extraviar o robar. En la actualidad, hay dos enfoques para asegurar dispositivos, contenidos y aplicaciones: Gestión de Dispositivos Móviles (MDM, del inglés Mobile Device Management) y Gestión de Aplicaciones Móviles (MAM, del inglés Mobile Application Management). Ambos han sido

combinados en lo que se conoce como Gestión de la Movilidad Empresarial (EMM, del inglés Enterprise Mobility Management).

MDM ofrece a la empresa un control total sobre un dispositivo, por lo que cuando un dispositivo se pierde, se roba o un empleado se marcha llevándoselo, la empresa puede limpiar el dispositivo y esencialmente hacer que este deje de funcionar. Este enfoque por lo general se toma cuando la empresa es propietaria del dispositivo y por lo tanto todos los datos y aplicaciones en el dispositivo son de su propiedad: todos los datos personales almacenados en el dispositivo se almacenan en él a riesgo del empleado.

MAM permite a una empresa adoptar BYOD, ya que hace que la empresa pueda asegurar aplicaciones y contenidos descargados a un dispositivo sin quitarle el control del mismo al propietario. Cuando un empleado deja la empresa llevándose consigo su dispositivo, la compañía puede deshabilitar las aplicaciones empresariales y limpiar cualquier contenido descargado en el dispositivo sin afectar a los datos personales, como fotografías y aplicaciones de consumo compradas por el empleado. La mayoría de soluciones de MDM y MAM son multiplataforma, siendo compatibles con múltiples dispositivos, y esto siempre debe tenerse en cuenta a la hora de decidirse por un proveedor de MDM o MAM.

Hay varias funciones de seguridad disponibles a través de estas dos soluciones de gestión, incluyendo las siguientes:

- Monitorización de dispositivo
- Control de licencias
- Distribución vía solución interna over-the-air (OTA)
- Inventariado de software
- Control de recursos
- Control remoto
- Gestión de la conexión

— Soporte y distribución de aplicaciones

Las medidas de seguridad incluyen entre otras:

— Protección de contraseñas
— Encriptación de datos en el dispositivo
— Encriptación de datos OTA
— Bloqueo remoto de dispositivos
— Limpieza de datos remota
— Reaprovisionamiento de dispositivos
— Copia de seguridad de datos en los dispositivos

Ejemplos de proveedores de EMM:

— **Airwatch**: *air-watch.com*
— **App47**: *app47.com*
— **Apperian**: *apperian.com*
— **Good**: *good.com*
— **Microsoft**: *microsoft.com/en-us/windows/windowsintune*
— **MobileIron**: *mobileiron.com*
— **Mocana**: *mocana.com*
— **SAP Afaria**: *go.sap.com/product/technology-platform/afaria-mobile-device-management*

Plataformas de Desarrollo de Aplicaciones Móviles (MADP)

Por lo general, un elemento clave de las aplicaciones empresariales es la sincronización de datos. Los dispositivos móviles tienen que recibir información actualizada o relevante de los servidores corporativos y los datos actualizados o recogidos tienen que ser enviados de vuelta. El nivel de acceso a los datos está determinado por las responsabilidades del usuario, así como por la política de confidencialidad. En cualquier caso, la sincronización tiene que ser segura, ya que los datos de las empresas son uno de sus activos más preciados. Además, una aplicación integrada en toda la empresa debe ser multiplataforma. Para compensar las deficiencias de los SDKs nativos así como de las soluciones multiplataforma más comunes en este aspecto, es posible que desees evaluar las soluciones tipo Plataformas de Desarrollo de Aplicaciones Móviles (MADP, del inglés Mobile Application Development Platform). Las MADP son entornos de desarrollo móvil que proporcionan el software intermedio y las herramientas para desarrollar, probar, implementar y administrar aplicaciones empresariales que se ejecutan en múltiples plataformas móviles con distintas

fuentes de datos en el back-end. Su objetivo es simplificar el desarrollo y reducir sus costes, en entornos donde las habilidades técnicas se deben mantener para múltiples plataformas, herramientas y cuestiones complejas, como la sincronización de datos y la autenticación.

Algunas soluciones disponibles incluyen las siguientes:

- **IBM MobileFirst Platform**: *www.ibm.com/mobilefirst*
- **KonyOne**: *www.kony.com/products*
- **Pega Amp**: *www.pega.com*
- **SAP Mobile Platform**: *www.sap.com/smp*
- **Spring Mobile Solutions**: *www.springmobilesolutions.com*

Seguridad en Aplicaciones Empresariales

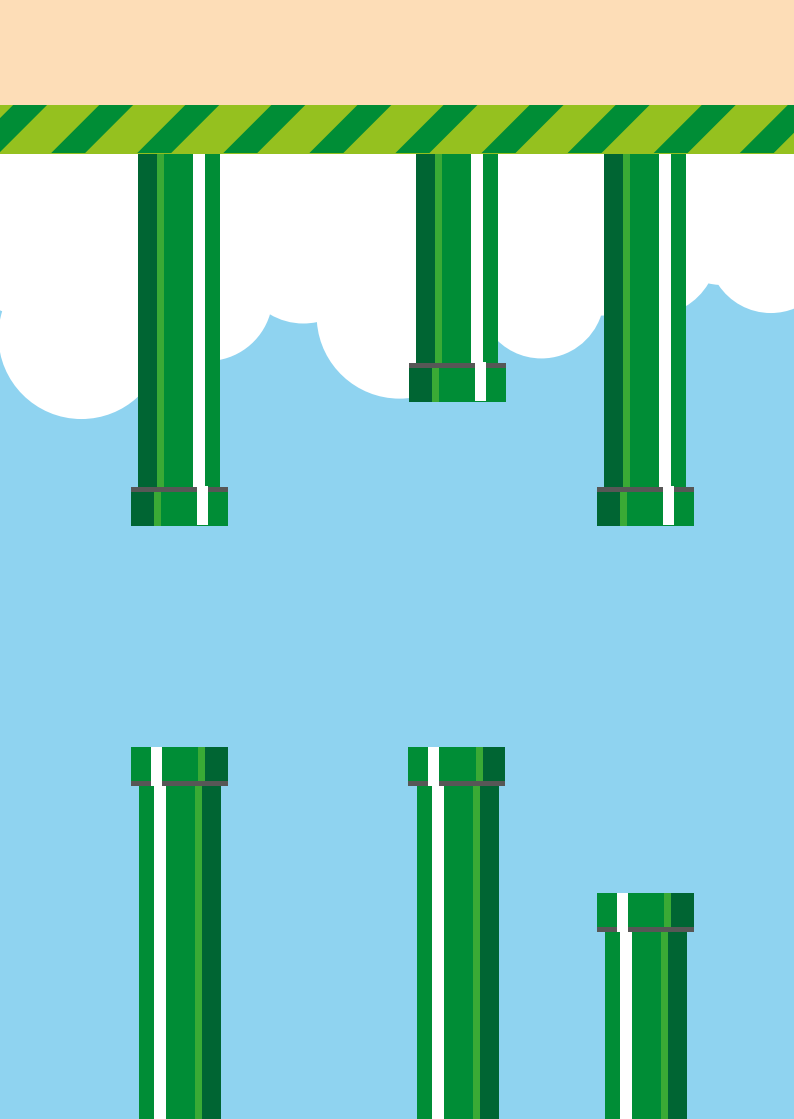
Una de las funciones principales de cualquier departamento de TIC es asegurarse de que todos los aspectos de la infraestructura de la empresa están asegurados contra ataques, de modo que no haya fugas de datos y estos no se vean comprometidos o robados. Dado que los dispositivos móviles son una extensión de la infraestructura TIC de una empresa, todas las aplicaciones empresariales deben ser diseñadas para asegurarse de que no se puedan utilizar para obtener acceso ilegal a la red interna corporativa. Como desarrollador de aplicaciones empresariales, por lo general se te pedirá que te ajustes a las normas que la empresa haya establecido en sus políticas de seguridad, así que debes estar preparado para responder a preguntas acerca de la seguridad de tu aplicación, como el cifrado de datos, la comunicación en red y la manera de lidiar con dispositivos jailbreakeados o rooteados.

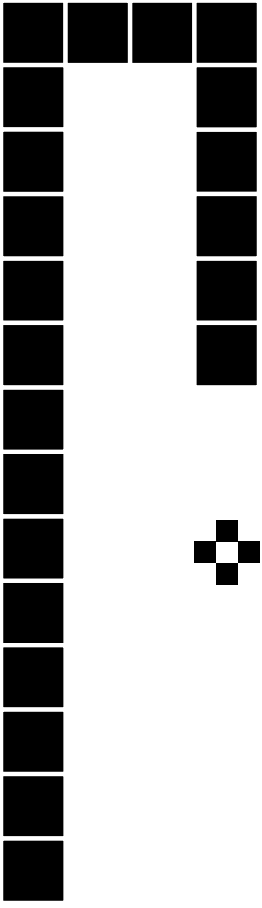
Muchos proveedores de EMM intensifican la seguridad en aplicaciones utilizando técnicas como el envolvimiento de

aplicaciones o proporcionando un SDK a los desarrolladores. Estas funcionalidades y las actualizaciones regulares de las plataformas permiten a las empresas bloquear de manera remota sus aplicaciones y mantenerse al día en cuanto al voluble panorama de la seguridad móvil sin tener que invertir demasiado esfuerzo y dinero en esta tarea.

Puntos Clave para Asegurar Aplicaciones Empresariales

- Si usas un proveedor de EMM, verifica que tenga las funcionalidades en seguridad suficientes para dar soporte a tus estándares corporativos.
- Cuando almacenes un dato en el dispositivo, asegúrate de que está encriptado.
- Cuando comuniques con un servicio web, usa siempre https.
- Además de lo anterior, asegúrate de que realizas verificaciones de puntos finales tanto en la aplicación como en el servicio web, para confirmar que tanto el servidor como el dispositivo son válidos.
- Siempre verifica que los ajustes del dispositivo cuentan con una suma de comprobación para asegurarte de que los valores no pueden ser cambiados una vez instalada la aplicación en el dispositivo.
- No permitas que la aplicación se ejecute en dispositivos jailbreakeados o rooteados.
- Ten un método para deshabilitar la aplicación si esta detecta que ha sido comprometida.
- Asegúrate de que el uso de encriptación es conforme a normativas de exportación y legislación relevantes para la región donde se utiliza la aplicación.





Juegos para Móviles

La Economía de los Juegos para Móviles

Antes de comenzar a hablar sobre el desarrollo de juegos para móviles deberíamos tratar de entender qué está impulsando ese mercado. El auge de los videojuegos móviles desde los primeros días de los juegos desarrollados en Java (técnicamente J2ME) sigue siendo asombroso. La firma de investigación del mercado de videojuegos Newzoo elevó recientemente sus estimaciones según las cuales el mercado global de los juegos de móvil superará los 40.000 millones de dólares en 2017[1]. Según VentureBeat[2] los juegos representan solamente el 40% de todas las descargas de aplicaciones en tiendas pero alrededor del 75% del gasto total, estando la mayoría de esos ingresos centrados en los 10 juegos líderes, en su mayoría F2P (free-to-play), es decir, total o parcialmente gratuitos. Algunos de estos juegos han dominado los rankings durante los últimos dos años. Juegos como Clash of Clans, Candy Crush y The Simpsons Tapped Out han generado miles de millones y han puesto los juegos móviles al menos al mismo nivel que el mercado de videoconsolas.

Es importante saber que para iOS y Android la gran mayoría de los ingresos de los juegos para móviles provienen de juegos F2P. También vale la pena señalar que el 7% de los ingresos

1 www.applift.com/blog/mobile-games-market-update.html

2 venturebeat.com/2014/11/04/candy-crush-leads-in-u-s-and-u-k-but-clash-of-clans-reigns-in-mobile-crazy-south-korea

en iOS proviene de juegos de alta calidad, casi el doble que en Android[3].

Hacer que los juegos funcionen en múltiples plataformas es cada vez más fácil. Alrededor del 54% de todos los juegos móviles están diseñados utilizando motores de terceros y el 45% de estos utilizan Unity. Hay muchos otros motores como Cocos, Corona, GameMaker, Unreal, etc. Cada motor ofrece diferentes ventajas y perspectivas para que desarrolladores con diferentes habilidades puedan dar forma a sus ideas y prepararlas para sacarlas al mercado rápidamente. Consulta el capítulo sobre multiplataforma de esta guía para obtener más información sobre los entornos disponibles.

Muchos desarrolladores viven bajo la ilusión de que serán el próximo desarrollador independiente que se hará rico. Esto ocurre con juegos como Flappy Bird, que ganan un increíble número de usuarios, pero estos casos están más cerca de ser premios de lotería que modelos a seguir. En cambio, es importante que los desarrolladores se den cuenta de que el mercado de juegos para móviles se ha convertido en un espacio sofisticado con muchas facetas y desafíos. Antes de empezar a crear tu juego necesitas prestar atención para comprender la naturaleza del mercado y de la gente. Una parte esencial de esto es que se ha convertido en un mercado enormemente competitivo, con un gran número de pequeños equipos produciendo grandes volúmenes de contenido y gastándose millones en desarrollo y publicidad para conservar sus posiciones.

3 venturebeat.com/2014/04/25/apple-vs-google-a-world-view-on-the-mobile-gaming-war

Hacer el Juego Adecuado

La creación de experiencias agradables para nuestro público objetivo requiere tanta o más creatividad que nunca antes. Los jugadores esperan que creemos diversión y les mostremos nuevas ideas de contenido. Y también necesitan ver algo conocido a fin de poder comparar y que les ayude a entender y relacionarse con nuevos contenidos. Scott Rogers en su libro "Level Up" describió esto como "El Triángulo de la Rareza" ("Triangle of Weirdness")[4]. Afirmó que los juegos se componen de un mundo, actividades y personajes. Podemos cambiar cualquiera de estas cosas por nuevas ideas, pero no podemos cambiar las tres sin correr el riesgo de perder la audiencia.

El tipo de diversión que estamos buscando en los juegos es la que sucede cuando el jugador es capaz de suspender su incredulidad y participar en una experiencia que no tiene ningún valor en el mundo real. Llegamos a estar totalmente absorbidos en la mecánica y la narrativa de la experiencia. Curiosamente, el desafío y la frustración son tanto las motivaciones para jugar como las potenciales causas para abandonar la experiencia. Si somos capaces de mantener el equilibrio entre estos estados generaremos un estado de placer que todos los diseñadores de juegos conocen: el 'Flujo de Csikszentmihalyi'[5].

Tenemos que apreciar que lo que es divertido en el espacio móvil es diferente de otras plataformas e incluso puede parecer contradictorio. Necesitamos juegos que sean simples y accesibles, pero con suficiente profundidad, propósito y progresión como para retener la atención del jugador. Si nos fijamos en lo que ha sido un éxito y en lo que no, vemos que

4 mrbossdesign.blogspot.co.uk/2008/09/triangle-of-weirdness.html

5 scienceandvalues.wordpress.com/2010/02/26/csikszentmihalyis-flow-pleasure-and-creativity

un juego móvil tiene que darnos un éxito significativo en menos de un minuto, pero mantenernos jugando durante cientos de días. El juego tiene que conseguir que, en lugar de recorrer todo el contenido en una sola sesión, juguemos docenas de veces al día. Necesitamos un juego que sea familiar, pero que también destaque lo suficiente como para ser destacado por las tiendas de aplicaciones. Nuestro juego tiene que ser divertido (y a menudo gratis), pero aún así crear nuevas razones para que los jugadores quieran, o mejor todavía, necesiten gastar dinero en él. Y la lista de aparentes contradicciones continúa.

Hay juegos desarrollados a partir de mecánicas emergentes o bloques de construcción que se combinan creando resultados sorprendentes o estratégicos, como Chess o Clash of Clans. Luego están los construidos con una serie de puntos de decisión progresivos que se resuelven con pasos encadenados para crear una historia como FTL o Monkey Island. Incluso podemos crear juegos que incorporen la creatividad del jugador como Createria o Minecraft, o simples puzles abstractos como Threes o SuperHexagon. Sea cual sea el camino que tomemos, el equilibrio está en el centro de nuestro razonamiento como diseñador. Tenemos que decidir hasta qué punto el juego se verá afectado por la habilidad y cuánto por la suerte, el grado en que el juego sigue una narrativa fija o liderada por el jugador, y por supuesto la complejidad de los sistemas internos, ya sea sobre el desarrollo de personajes o una economía de recursos. En el F2P también tenemos que considerar el impacto del dinero gastado en la experiencia de juego.

Lo que a menudo olvidamos es tomar en consideración lo que le importa al jugador. Una de las preguntas más importantes que debes hacerte es por qué deben elegir tu juego para distraerse. Seamos sinceros, la mayoría de los videojuegos móviles comienzan como una distracción, incluso si al final nos pasamos más tiempo con nuestros teléfonos que con

nuestras consolas. Así que, ¿por qué habrían de jugar tu juego? Decir que es un buen juego no es suficiente. Tenemos que ser capaces de responder a esa pregunta con honestidad. Aprender del diseño clásico y del marketing de productos te ayudará a tomar buenas decisiones al respecto.

Enganchar al Jugador Móvil

Cuando desarrollamos juegos para móviles estamos creando una experiencia para entretener a los jugadores en un tipo específico de dispositivo. Tablets y teléfonos satisfacen necesidades diferentes y requieren una atención al detalle específica respecto a las necesidades de uso que cubren. En el teléfono todo gira alrededor del 'siguiente minuto', lo sacamos cuando esperamos que ocurra algo o necesitamos entretenernos. ¿Cómo usan los jugadores sus dispositivos tablet? Se trata más bien de un período de descanso o relajación más largo. ¿Qué implica eso para el juego que queremos crear?

La producción de nuestro juego es la combinación de una visión única, una narrativa de juego convincente y una experiencia diseñada para afectar a las emociones de los jugadores. Todo esto tiene que encajar con la forma en que se consume el juego y en el móvil es preciso comprender las restricciones inherentes a estos dispositivos. La forma en que los jugadores interactúan con el dispositivo se ve afectada por el tamaño limitado de la pantalla, los controles en una pantalla táctil, acelerómetros, duración de la batería, la posibilidad de ser interrumpidos, la facilidad que los jugadores tienen para salir del juego y guardar el dispositivo, la calidad limitada del altavoz, la salida de auriculares de alta calidad, etc. Los teléfonos móviles están (en su mayoría) conectados a internet y son el dispositivo más influyente, ya que siempre los llevamos encima.

Como ejemplo, piensa en la forma en que implementamos los controles. Las pantallas táctiles permiten una amplia gama

de movimientos en un plano 2D, pero al cabo de poco tiempo nuestra piel se calienta y pierde capacitancia, lo que significa que los controles serán menos fiables. Si simplemente tratamos de emular un sistema de joystick (como hacen demasiados juegos) pronto tendremos problemas con la experiencia de juego. Sin duda, esta es una de las razones por las que los juegos de disparo en primera persona no han sido tan populares en el móvil. En su lugar debemos diseñar la mecánica de juego con controles que hagan el contacto cómodo o que comprendan los límites de los métodos disponibles y los hagan parte de la experiencia. MiniGore es un buen ejemplo de juego en el que la dificultad se incrementa por culpa de la mecánica tipo joystick, que lo hace progresivamente más difícil de controlar. Por otro lado, Hayday de Supercell demostró de qué manera lo táctil podía ser absolutamente maravilloso. El movimiento utilizado para recoger las cosechas es tan agradable que eleva este juego muy por encima de otros juegos de granjas en cualquier plataforma, y resulta perfecto para una experiencia en tablet.

La necesidad de satisfacción inmediata de los que juegan en el dispositivo móvil no reemplaza la capacidad de enganchar a más largo plazo. Sin embargo, refuerza la necesidad de simplicidad en el juego, tan necesaria en las plataformas móviles. Pero la simplicidad por sí misma no podrá seguir manteniendo el nivel de interés en el juego. Para mantener a la gente jugando necesitamos crear un contexto que nos dé una razón para repetir la mecánica de juego. Algo que nos dé un sentido de propósito y progresión, que nos llame la atención después de que la sesión de juego haya terminado, animándonos a volver a jugar. Conviene que esa mecánica inicial sea agradable incluso tras miles de horas de juego.

Juegos como CSR y Candy Crush introdujeron este método de diseño de juegos en el mercado móvil. Hicieron algo mágico: encontrar maneras de encadenar una serie de procesos

mecánicos para mantener la jugabilidad a través de miles de sesiones, al mismo tiempo que se da la sensación de que las metas se consiguen. Esto construye un compromiso a largo plazo manteniendo a los jugadores involucrados en tu juego durante más tiempo, siempre y cuando la experiencia sea suficientemente significativa. Conseguir que los jugadores estén más tiempo jugando tiene un impacto directo en su disposición a gastar dinero en el juego. En una encuesta realizada en el 2014 por Unity[6], el gasto medio de los jugadores que pasan menos de una hora con un juego es de 0,66 dólares, pero para los que están más de 10 horas se eleva a 15,15 dólares.

Diseñar el Trayecto del Jugador

Comprender la importancia del vínculo a largo plazo con el jugador tiene un impacto profundo en el diseño de videojuegos. La idea de un juego como una mecánica o una historia se transforma cuando se comprende que no es solamente el héroe del juego el que se embarca en un trayecto, sino que nuestro jugador también lo hace. Este trayecto está formado por varias etapas:

1. Descubrir

Los jugadores tienen un conjunto particular de necesidades y aspiraciones cuando se encuentran con tu juego por primera vez. Tenemos que asegurarnos de que el camino hacia este descubrimiento establece condiciones propicias para que se descarguen y jueguen el juego por primera vez. Por lo general hay pocas oportunidades de establecer las expectativas correctas, pero hacerlo es esencial. Si el juego tiene un coste inicial, informa al jugador de por qué debería comprarlo y lo que va a perderse si no lo hace. Si el juego es gratuito también tenemos que crear expectativas, y además debemos mostrar al jugador por qué la publicidad supone un mal menor. O, si se pueden realizar compras

6 www.gamesindustry.biz/articles/2014-10-14-mobile-spending-driven-by-35-44-year-olds

desde la aplicación, por qué vale la pena realizarlas. Este es un arte delicado. Debes ser claro y transparente, y aún así comunicar por qué el juego es digno de su tiempo e inversión.

2. Enganchar y Subir a Bordo

Una vez que el jugador ha tomado la decisión de instalar el juego tenemos que hacer tan fácil como sea posible que se involucre con él. Haz el icono y nombre del juego inmediatamente reconocibles e, idealmente, retadores. Que supongan una razón para iniciarlo. En esta etapa no queremos que tome decisiones acerca de qué personajes utilizar o qué niveles elegir (todavía no sabría hacerlo). No le hagas iniciar sesión en Facebook o configurar una cuenta antes de jugar, simplemente muéstrale de qué trata el juego. Después sorpréndele. Podemos comparar esto a "La Apertura Bond": la primera vez que se juega a un juego debe ser como los primeros 5 minutos de cada película de James Bond. Nos impresiona y al mismo tiempo establece todo lo que necesitamos saber acerca de la historia, los agentes supersecretos y el escenario en el que tiene lugar. Pero hace más que eso: asegura que no nos queramos levantarnos del asiento. Es esta doble función de mostrar sin necesidad de explicar lo que fija las expectativas para el resto de la película, algo que también se aplica a los videojuegos. En un juego, más que mostrar o explicar lo que queremos es '¡hacer!'. El jugador tiene que aprender acerca de los juegos mediante la acción y disfrutando de sus logros rápidamente.

3. Construir el Andamio

Si tenemos éxito y logramos instruir a los jugadores y establecer las expectativas correctas tenemos que mantenerles jugando. Ten en cuenta que necesitan razones para volver. Para que un juego se convierta en algo que jugamos regularmente y durante mucho tiempo, tenemos que entender cómo se percibe el éxito en el mismo. Debemos saber que habrá retos y progresión, así como un sentido de propósito. Esto significa

que tenemos que vernos fracasar y aún así querer continuar jugando. Necesitamos pensar que tenemos asuntos pendientes que nos obligan a superar la dificultad de volver a jugar. Hay que entender las razones para seguir y tener una serie de objetivos alcanzables, sin que compliquen en exceso el juego. Una técnica que puede ayudar es el concepto del momento culminante.
En él se tienen en cuenta dos cosas. En primer lugar, estamos aceptando que tenemos que dar a nuestra audiencia una interrupción natural en la experiencia de juego, eso es importante. No estamos diciendo que haya que evitar que los jugadores continúen, pero sí decimos que darles la opción de detenerse es útil para construir el compromiso, siempre y cuando se les dé una razón para volver. Esa razón para volver podría ser esperar para tener más combustible, para que las plantas crezcan, para que un vehículo esté reparado o incluso para que otro jugador visite la ciudad. La clave es crear la sensación de que si no vuelven se estarán perdiendo algo. Pero sin ser penalizados. Si eres capaz de animar a los jugadores a querer regresar con regularidad y crear "citas" para jugar, entonces sabrás que están verdaderamente "enganchados".

4. Final del Juego

Una vez que los jugadores están verdaderamente comprometidos, hay otros factores que se vuelven importantes si quieres mantener su interés. Necesitas tirar el juego hacia adelante con necesidades cambiantes. Por ejemplo en forma de contenido o de extensiones del juego. También podría ser a través de **juego social** creando una comunidad, unos lazos de unión y un sentido de pertenencia mediante el establecimiento de reglas compartidas. Cuando diseñamos juegos tenemos que pensar en cómo las reglas que utilizamos para entretener y crear un compromiso también proporcionan los medios para que las personas se identifiquen con nuestro juego y lo expresen. Los factores sociales son increíblemente importantes y pueden tener

un impacto notable, no solo en términos de participación sino también en los ingresos. Si una compra dentro de la aplicación no solo ayuda a sacarle más partido al juego sino que también hace ganar prestigio ante otros jugadores, los jugadores serán más propensos a hacerlo.

Cabe destacar que las comunidades sociales ayudan a crear un compromiso más profundo que algunos han denominado 'Ballenas' pero que también podemos definir como "auténticos fans". Se trata de los jugadores que gastan cantidades significativas en un juego y que a menudo pueden ser tu fuente principal de ingresos. En muchos juegos estos jugadores no serán especialmente sociables, ya que preferirán mejorar su capacidad de juego en lugar de relacionarse con otros. Sin embargo, la presencia de otros jugadores crea condiciones favorables que permiten que surjan estos jugadores. Sin jugadores en modo gratuito tendemos a no conseguir el nivel de compromiso de los auténticos fans. Es importante no confundir estos jugadores con las personas que son adictas. En la adicción los individuos tienen una compulsión que abruma su comportamiento que de otra manera sería racional. En la práctica, la mayoría de los auténticos fans son personas racionales que han hecho de tu juego su principal afición. Una conducta adictiva siempre es perjudicial para el individuo y debemos hacer todo lo posible para ayudar a cualquier persona con este tipo de problemas.



5. Abandono

La etapa final de la vida del juego es el "churning" o abandono. Es inevitable que al final los jugadores dejen de jugar a nuestro juego. Queremos retrasar este momento el mayor tiempo posible, pero ignorar este hecho y no planificarlo nos dará problemas. Para entender lo que mantiene a la gente jugando durante un período de tiempo prolongado piensa en el "Giro Colombo". Se basa en la serie detectivesca en la que el actor Peter Falk hace del torpe teniente con el mismo nombre. Lo que hizo de esta una de las mejores series de televisión de todos los tiempos es que hizo algo extraño con la narrativa de un misterioso asesinato. El espectador ve quién lo hizo. ¿Dónde está el atractivo de una serie sobre asesinatos si se sabe quiénes los cometen? La clave del espectáculo estaba en esperar el momento en que Colombo decía esas famosas cuatro palabras: "¡Solo una cosa más!", cosa que siempre ocurría en los últimos minutos del episodio. El detective habría estado hablando por cuarta o quinta vez con el asesino sobre alguna aparentemente aleatoria cuestión. Entonces suelta esa frase y el espectador sabe que en ese momento procederá a aclarar todo lo referente al asesinato. No solo quién lo hizo, sino por qué y cómo Colombo lo había descubierto. Esa es la clase de recompensa predecible que el espectador estaba esperando. ¿Qué es lo que hace que los jugadores vuelvan a tu juego incluso cuando saben todo lo que va a ocurrir?

Analíticas y Flujo del Juego

El diseño del juego desde la perspectiva de las etapas de vida del jugador es muy importante, pero a fin de aprovechar los beneficios de esta estrategia necesitamos saber de qué manera responde el jugador a esa experiencia, y eso significa que necesitamos conocer la actividad del jugador en nuestro juego. Por ejemplo, podríamos querer conocer que un jugador recibió un disparo en un juego FPS (del inglés "first-person shooter", juegos de disparar en primera persona). Podríamos darle un nombre a ese caso, como por ejemplo 'JugadorDisparado' y almacenar una serie de variables con ese evento. Esto podría incluir la fecha y la hora en que se produjo, un identificador anónimo del jugador, las coordenadas 'X,Y,Z' de su avatar, el daño recibido y un identificador anónimo del jugador que le disparó, así como un identificador de la sesión del juego.

Es importante destacar que no tenemos que capturarlo todo. Hay algunos tipos de datos que son información estática de referencia. Por ejemplo, la posición específica en un mapa específico. Siempre y cuando la versión del mapa utilizado en ese momento sea conocida, bastará con las coordenadas X,Y,Z para crear un mapa de calor posterior. También podemos inferir una gran cantidad de datos de otros eventos, siempre que exista alguna información de conexión. Por ejemplo, no es necesario capturar el nivel que el jugador está jugando en cada evento o la lista de todos los jugadores en esa sesión. Podemos capturar esa información en una serie de eventos específicos 'Iniciar sesión' y utilizar el identificador de sesión asociado para que nos permita identificar todo lo que pasó en esa sesión específica del juego.

Pero incluso si quisiéramos capturarlo todo, nunca seríamos capaces de hacerlo: los datos que recojamos siempre serán incompletos. Por ejemplo, si la batería se agota o el jugador cambia de aplicación para recibir una llamada telefónica probablemente no recibiremos los últimos datos. Esto es menos problemático en un juego basado en un servidor, pero incluso así nunca se llega al 100% y el aspecto negativo es que el juego no se puede jugar sin conexión a internet, perjudicando nuestra oportunidad para que se creen hábitos de juego.

Nótese que queremos que nuestros jugadores se mantengan en el anonimato. No queremos ni necesitamos espiar a nuestros jugadores, sino que necesitamos entender cómo funciona el juego para todos los jugadores.

¿Qué Eventos Deberíamos Capturar?

Al considerar de qué eventos realizar un seguimiento, piensa en ellos en términos de la línea de tiempo en la que el jugador podría encontrárselos. Existirá, por supuesto, el caso del que juega por primera vez, pero también puede ser útil mapear la experiencia y el flujo de sesión de los jugadores más experimentados. No se trata de mapear necesariamente cada botón que se aprieta, sino de mirar los momentos en los que se toman decisiones significativas. Hay un enfoque utilizado por la industria de la alimentación llamado HACCP[7].

7 develop-online.net/opinions/navigating-the-hazards-of-game-data/0187815: en esencia, el tema es que estamos buscando los "peligros", como por ejemplo si se abandona el juego, pero también los puntos que desencadenan acciones más positivas, como pagar por una compra dentro de la aplicación o mirar un vídeo con publicidad.

Algunos eventos típicos que merece la pena monitorizar incluyen los siguientes:

- **GameMenuLaunch:** AnonPlayerID; TimeIconLaunched
- **SessionLaunch:** TimeSessionLaunched; AnonPlayerID(s); SessionID; LevelIDSelected; OptionSelected
- **SessionStart:** TimeSessionStarted; AnonPlayerID; SessionID;
- **ObjectiveSet:** TimeObjectiveSet; AnonPlayerID; SessionID; ObjectiveID;
- **ObjectiveMet:** TimeObjectiveMet; AnonPlayerID; SessionID; ObjectiveID; Score; Reward; XYZLocation
- **TargetHit:** TimeTargetHit; AttackerID(AnonPlayerID?); SessionID; TargetID(AnonPlayerID?); Damage, XYZLocation
- **PlayerDeath:** TimePlayerDeath; AnonPlayerID; SessionID; XYZLocation
- **LevelComplete:** AnonPlayerID; SessionID; ObjectiveID; Score; Reward; XYZLocation

Creando eventos de esta manera podemos inferir una enorme cantidad de información. Por ejemplo, si queremos conocer el porcentaje de jugadores que completan un nivel podemos comparar el número de eventos 'GameMenuLaunch' con el de eventos 'LevelComplete'. Pero todavía podemos ir más allá con nuestro análisis. Podemos mirar cuántas personas completaron un ObjectiveID concreto en un 'LevelIDSelected' específico y compararlo con el número de 'LevelComplete' del siguiente nivel para averiguar si saltarse objetivos en los niveles anteriores tienen un especial impacto en el rendimiento posterior.

La captura de datos de esta manera puede ser muy potente, sin embargo la manera de capturar la información estará inherentemente sesgada por la manera en que crees que el juego se supone que se juega. Esto significa que tenemos que

revisar constantemente nuestras métricas. El mayor problema es que no podemos capturar lo que los jugadores 'harán' o 'habrían hecho'. Esto puede parecer obvio, pero vamos a utilizar las analíticas para tomar decisiones de diseño informadas. Si es cierto que solamente el 2% de los jugadores gastan dinero en un juego F2P, eso significa que no sabemos lo que habría impulsado al restante 98% a gastar. Puede que no haya nada que les hubiera convencido pero el punto a destacar es que la captura de datos nunca puede ser completa y eso significa que tenemos que considerar la significación estadística y ser muy cautelosos a la hora de asumir causalidad en lugar de correlación. Mirar el ciclo de vida de un jugador nos ayuda a mitigar esto porque podemos descomponer cada etapa de decisión y buscar formas de aumentar la probabilidad de convencer jugadores en cada etapa. Todo consiste en hacer las preguntas correctas en el momento adecuado.

Gratuitos versus de Pago

Las discusiones entre los juegos gratuitos y de pago se han convertido casi en tribales entre los desarrolladores de juegos, preguntándose si los modelos de negocio han empañado la naturaleza del diseño de videojuegos o incluso haciéndose preguntas acerca de la moralidad de estos diseños centrados en el dinero.

En términos económicos, está claro lo que sucede cuando la oferta aumenta: los precios caen. Con un suministro que a efectos prácticos es infinito, el precio cae hasta cero. Esto es exactamente lo que ha sucedido y la explicación de que el modelo F2P esté dominando. ¿Pero qué ocurre con ese 7% de los ingresos en iOS que procede de juegos premium? Como en cualquier mercado, cuando tratamos con competidores tenemos una opción. Podemos o bien buscar volumen (crear un producto básico) o diferenciarnos (crear un nicho). Los juegos premium

de éxito son los que han sido capaces de atraer a una audiencia ofreciendo algo que es percibido como de mayor valor que el resto de los juegos disponibles. Juegos como Monument Valley o The Room han demostrado que esto es todavía posible y se les descubre debido a su precio de venta. De todos modos, no han llegado a la misma línea de ingresos que los juegos F2P que mejor han funcionado, a pesar de la insistencia de las tiendas de aplicaciones por potenciarlos.

El movimiento hacia lo gratuito no ha sido un camino fácil para muchos desarrolladores. Y los intentos de 'clonar' los modelos de negocio de juegos como Clash of Clans o Candy Crush rara vez han visto un nivel comparable de éxito. Todo esto a pesar de que la fórmula aparenta ser muy sencilla: tomar una mecánica simple de juego y ofrecer un nuevo contexto social con el fin de que sea infinitamente repetible; a continuación, añadir una forma de fricción que dificulte que el jugador alcance sus objetivos, pero que esté programada para dejar el objetivo siempre 'en la punta de los dedos'; finalmente permitir que la gente pague para eliminar esa fricción, pero hacer que cada objetivo se convierta en el detonante de uno nuevo, también fuera de su alcance por poco.

Por supuesto todo esto no es tan fácil y este tipo de fórmula es algo que puede convertirse rápidamente en 'bastante poco divertido' y generar una gran cantidad de abandonos. Incluso si el desarrollador añade una buena cantidad de análisis de datos para averiguar dónde están abandonando los jugadores o para encontrar la mejor manera de que paguen más, al final el juego muere inevitablemente. Peor que eso, a medida que más juegos se perciben con esa aplicación rígida de un modelo de negocio más allá del disfrute del propio juego, más jugadores rechazan ese tipo de juegos. Si percibes que el juego no es más que un intento de sacarte el dinero, ¿crees que estarás muy predispuesto como jugador?

Lo que hace que esos juegos funcionen es que comprenden a las personas y saben cómo construir un servicio que les permita sentirse competentes y en control de la situación y ser capaces de escapar de sus vidas cotidianas. Los beneficios llegan cuando podemos extender lo más posible el deleite que sienten los jugadores y su compromiso a largo plazo, dándoles una razón para querer gastar dinero. Necesitamos crear valor durante mucho tiempo, no solo ingresos a corto plazo. Tenemos que encontrar una manera de involucrar al jugador con el contenido que estamos creando para su entretenimiento.

Un juego F2P bien hecho saca el valor del jugador gratuito no solo en términos de su potencial viral, sino también en términos de creación de una marca y condiciones que animan a los jugadores a pagar. Sin embargo, los juegos F2P no tienen una gran reputación, especialmente entre los padres preocupados por las compras dentro de la aplicación y entre algunos diseñadores de juegos. Ha habido algunos escándalos notables, en especial con niños comprando 'accidentalmente' miles de dólares en compras dentro de la aplicación. Este argumento ha empañado la imagen de la industria de los juegos en su conjunto y ha atraído la atención de los reguladores legales, incluida la UE, que han publicado directrices sobre la venta dentro de las aplicaciones enfocadas a niños. Sin embargo, hasta la fecha la regulación parece haber sido comprensiva y lógica. Con toda razón, han pedido a diseñadores y distribuidores que comuniquen qué se vende y cómo se accede a ello, en particular en el caso de un público menor de edad. Se ha solicitado a las plataformas que realicen importantes cambios, aunque de bajo impacto, en su funcionamiento (aunque de todos modos esto ya estaba pasando). En resumen, les han pedido aclarar las expectativas de uso que ya están impuestas por la legislación vigente.

Los diseñadores de juegos F2P suelen hablar del condicionamiento operante, y en particular de un experimento conocido como 'la caja de solución de problemas' o 'skinner box'. Estas cajas, que llevan el nombre del psicólogo que las creó, permiten a animales obtener alimentos pulsando un botón. El experimento descubrió que la variación de la velocidad a la que el botón liberaba la comida afectaba al comportamiento de los animales. La premisa es que en los juegos los diseñadores encuentran la manera de recompensar a los jugadores y que este es un método equivalente de condicionamiento. Es cierto que algunos experimentos han demostrado que ese tipo de condicionamiento puede funcionar en los seres humanos, al menos a corto plazo, pero en los juegos no controlamos todos los estímulos ni utilizamos alimentos (u otras necesidades de bajo nivel) como motivación. Como diseñadores, disponemos de métodos mucho más potentes para retener a un jugador que dar recompensas a corto plazo. Podemos contar historias. Podemos deleitarlos con estímulos visuales y sonoros. Podemos crear juegos que se convierten en experiencias sociales compartidas. Todas estas opciones tienen un efecto estimulante sobre los usuarios mucho mayor que cualquier ejercicio de condicionamiento operante. En el corto plazo es posible (pero no deseable) manipular a los jugadores, especialmente en el caso de personas vulnerables como los niños, pero si lo haces esa manipulación no durará mucho tiempo y a largo plazo reducirá nuestro valor del ciclo de vida. A nivel comercial es mucho más eficaz hacer buenos juegos.

El concepto de adicción es algo que siempre hemos utilizado como atajo para decir que un juego es bueno y atractivo. Ahora se ha convertido en una fuente de preocupación. Cualquier juego recompensa a los jugadores con una liberación de dopamina cuando tienen éxito, lo cual crea una respuesta fisiológica no muy diferente al consumo de drogas o el ejercicio físico. Sabemos que en el caso de los juegos de azar esta

clase de química en el cuerpo puede crear adicción, que como hemos comentado anteriormente es una compulsión que anula el comportamiento racional. Sin embargo, los juegos de azar ofrecen un estímulo muy diferente al del resto de juegos. Con los juegos de azar el estímulo se basa en la incertidumbre y en la apuesta que hacemos. Esa incertidumbre es constante cada vez que jugamos. Con los juegos, en cambio, aprendemos la mecánica y eso reduce la potencia del estímulo con el tiempo. Aunque la adicción a los videojuegos es un problema reconocido, es algo que pertenece al ámbito de la adicción conductual, donde una actividad normal se convierte en compulsiva. Este tema aún está en investigación y a día de hoy no está incluido en el Manual diagnóstico y estadístico de los trastornos mentales, en su quinta edición (DSM-5).

Compra desde la Aplicación

La forma más fácil de entender el modelo F2P es darse cuenta de que estos juegos simplemente han llevado el lado minorista al interior del juego. Eso significa que las personas que conocen mejor el juego y a sus jugadores (es decir, los desarrolladores) pueden identificar los elementos que los jugadores desearán y que complementen y enriquezcan la experiencia para todos los jugadores; y así vendérselos directamente al público que adora ese juego. Sin embargo, muchos de los juegos actuales se han centrado en la puesta en práctica relativamente simplista consistente en utilizar algún tipo de energía o sistema similar de 'bloqueo', de tal manera que no puedes seguir jugando hasta que pasa un tiempo o realizas una compra.

Esto puede ser muy insatisfactorio dependiendo del juego, pero afortunadamente no es la única manera de crear un sistema de monetización del juego y podría decirse que ha dado lugar a discusiones inútiles sobre la ética del modelo F2P.

No obstante un estudio surcoreano[8] reveló una verdad importante en lo que al comportamiento a la hora de comprar juegos se refiere: los jugadores no gastan dinero porque estén felices o porque les gusten los desarrolladores (tristemente). Gastan dinero porque esperan valor futuro.

Como resultado tenemos que saber lo que los jugadores necesitan en sus diferentes etapas de la vida y satisfacer eso. Al igual que cualquier decisión de compra, las ofertas dentro del juego tienen que superar las objeciones potenciales ('el comportamiento de los clientes como riesgo'). Los siguientes ejemplos muestran cómo intentan hacer esto algunos juegos.

— **Asuntos Pendientes**: Los juegos como Kim Kardashian Hollywood hacen un trabajo increíble con la progresión narrativa y el formato de lo que son esencialmente tareas consistentes en ir tocando elementos en la pantalla para crear una sensación entretenida de asuntos pendientes. La manera de jugar puede parecer limitada, pero el modo como engancha al jugador es de hecho muy real, dejándole siempre con ganas de más.
— **Relevancia Continua**: Los juegos como VEGA confict muestran artículos que los jugadores ya han desbloqueado y los que serán capaz de desbloquear más adelante en el juego. Hay una comparación de estadísticas y costes para demostrar la pertinencia de lo que ya hemos desbloqueado así como por qué deberíamos continuar jugando para desbloquear más.
— **Capital Social**: Demasiados juegos ignoran el impacto de la visibilidad social no solo a la hora de comprar sino también sobre los niveles de satisfacción y viralidad. Especialmente cuando los jugadores tienen la posibilidad de personalizar

8 staffweb.hkbu.edu.hk/vwschow/lectures/ism3620/c26.pdf

su experiencia. Esto fue clave para los ingresos de PlayStation®Home (ahora cerrado) con ejemplos como 'Gold Suit' ('traje dorado') que ofrecía a los que lo llevaban prestigio por el hecho de tenerlo.

Muchos juegos tienen una alta proporción del gasto teniendo lugar en las primeras horas en que el jugador inicia un juego, sin embargo esto es un desperdicio si esos jugadores no gastan más de una vez. Si esto sucede para la mayoría de nuestros jugadores que pagan, debemos considerar nuestro sistema de monetización como un fracaso. Como diseñadores siempre tenemos que considerar en qué modo un jugador que hace una compra hace que la experiencia sea mejor para todos y por qué lo harían de nuevo. Una y otra vez. Una y otra vez...

Los desarrolladores que buscan ofrecer un mayor nivel de utilidad (o de valor esperado) a los productos que venden en el juego están empezando a adoptar un enfoque más parecido al del diseño de juegos para los tipos de mercancías que ofrecen a los jugadores. Esto se puede clasificar en cuatro categorías:

1. **Sustento**: bienes que necesitamos para seguir jugando
2. **Atajos**: bienes que aceleran las acciones que estamos llevando a cabo
3. **Socialización**: bienes que son básicamente de capital social
4. **Estrategia**: bienes que abren nuevas opciones de juego

Estos bienes pueden venir en varios formatos:

- Consumible: un artículo de un solo uso
- Capacidad: algo que mejora el crecimiento o jugabilidad
- Permanente: una actualización permanente o el desbloqueo de un artículo
- Generadores: un incremento en el subministro de un consumible

Mirando el trayecto del jugador en tu juego, deberías ser capaz de identificar un punto en la mecánica del juego o en los bucles del contexto de juego (tal vez incluso en la estrategia del juego) donde puedas combinar estos tipos y formas. Por ejemplo, en un juego de conducción el combustible es un bien consumible/sustento; pero también puedes contemplar vender un aumento en el tamaño del depósito de combustible (un bien capacidad/sustento) u ofrecer una bomba de gasolina que entrega cierta cantidad de combustible por día (un bien generador). Así, cuando un jugador actualice a un coche mejor, parte de la estrategia de juego puede verse influenciada según el coche gaste más o menos combustible que otros coches disponibles.

De cara al éxito de las compras dentro de tu aplicación, es crítico no solo hacer que lo que vendes sea deseable sino también asegurarte de que puede escalar. Queremos que nuestros jugadores compren una y otra vez y esto solo ocurre cuando proporcionamos valor consistente y predecible pero sin que deje de ser sorprendente. Si tu juego tiene solo una manera de medir el éxito, ¿cómo podrás crear valor escalable? Considera un juego de disparar en primera persona, ¿si alguien va a gastar dinero, por qué debería comprar cualquier cosa que no sea el arma más grande y potente que se pueda permitir pagar? Pero cuando haga eso, cualquier compra previa de otra arma de fuego será redundante. ¡Sin embargo si las armas tienen beneficios distintos como perforadoras, áreas de alcance, silenciadores, etc. entonces existen razones para comprar un arsenal más amplio de armas!

Para más detalles sobre este tema, consulta este artículo en el sitio web de Unity[9].

9 blogs.unity3d.com/2015/06/23/design-driven-in-app-purchases-creating-sustainable-monetization

Publicidad

La publicidad en los juegos es una manera muy exitosa de complementar los ingresos de las compras en la aplicación. De hecho, para algunos juegos puede suponer el modelo principal de ingresos. Los anuncios tradicionales como los banners o los intersticiales funcionan presentando una capa superpuesta de algún tipo que interrumpe la experiencia de juego. Sin embargo algunos de los nuevos juegos más importantes en el último año han estado utilizando anuncios de vídeo opcionales incentivados. En este modelo, el jugador elige mirar el anuncio con el fin de obtener un pequeño beneficio en el juego.

Hay un par de factores que afectan a la predisposición de los jugadores para elegir activamente recibir anuncios. El contenido del anuncio en sí mismo debe ser relevante, por ejemplo mostrando a un jugador un anuncio de otro juego, y también debe ser corto, más o menos entre 15 y 30 segundos. De todos modos, por lo general el jugador tiene que sacar algún beneficio. Pueden ser un buen número de cosas, desde monedas o recursos en el juego a vidas extra. Estas recompensas tienen que ser relevantes para el jugador y también repetibles, ya que siempre tiene que haber una razón para que el jugador vea otro anuncio. Si se puede desbloquear un elemento importante o un nuevo nivel al ver un vídeo, eso definitivamente supone un gran valor, pero es poco probable que genere la escalada de visualizaciones necesaria para impulsar los ingresos suficientes para hacer que valga la pena. Del mismo modo, si ver vídeos desbloquea tantos potenciadores que desequilibran el juego, se habrán terminado los motivos para jugar.

Angry Birds Go utiliza los anuncios de vídeo para permitir que los usuarios consigan un impulso gratis, pero únicamente al comienzo de la carrera. Esto no se sobrepone a la experiencia del juego, pero sí que requiere que el jugador cambie su atención de la estrategia de juego a la reproducción del vídeo.

La motivación es clara: si quiero un impulso en mi próxima carrera lo puedo conseguir a cambio de mi tiempo viendo este anuncio.

Sonic Dash ofrece una opción para que el juego continúe después de 'morir', ya sea usando una ficha para 'revivir' o viendo un anuncio de vídeo. Curiosamente la duración del anuncio (alrededor de 15-20 segundos) también permite al jugador tomar un breve descanso antes de reanudar el juego, cosa que puede mejorar tanto el rendimiento como la motivación.

Y todavía otro enfoque puede verse en Crossy Road, donde se ofrece a los jugadores una pequeña cantidad de monedas extra a cambio de ver el vídeo. Estas monedas se utilizan a cambio de una serie de personajes aleatorios (que añaden diversión pero sin cambiar el modo de juego). De manera interesante, la naturaleza aleatoria de las recompensas hace que estas no canibalicen la compra de esos mismos personajes, ¡de hecho ha tornado la decisión de compra más fácil! Hipster Whale hizo otra cosa también interesante. Limitan lo a menudo que se pueden ver los anuncios a un marco de tiempo predecible, haciéndolos aún más apetecibles.

Lo que es importante es que la recompensa de ver el vídeo es intrínsecamente parte del juego en sí. No hay ningún incentivo por separado para descargar el juego anunciado, pero si deciden hacerlo será porque les habrá gustado el aspecto del juego. Eso es lo que al fin y al cabo proporciona los ingresos y al mismo tiempo refuerza su compromiso con tu juego.

Ser Descubierto

Si has seguido estas pautas, habrás puesto tu diseño de juego en las mejores condiciones para que se adapte a tu audiencia y, con suerte, te dará una oportunidad de competir en el mercado. Sin embargo, esto por sí solo no es suficiente. Tenemos que usar todas las vías de comunicación que podamos y por lo general estas requieren inversión. Todavía es posible tener éxito sin tener que gastar dinero en publicidad, pero ello implica tener el boleto ganador en una lotería mundial. Esto se aplica tanto a juegos como a cualquier otro tipo de aplicación móvil. Encontrarás algunos consejos de cómo comercializar tu software en el capítulo sobre monetización de este libro. Adicionalmente, puede que te interesen las siguientes estrategias específicas para videojuegos.

Que la prensa repare en tu juego puede ayudar, especialmente si participas en competiciones como Pocket Gamer's Big Indie Pitch[10] o Indie Awards at Casual Connect[11]. También puede ayudarte atraer la atención de los usuarios más populares de YouTube.

Gastar dinero en publicidad puede ayudar, pero es importante ser consciente de que estás compitiendo con mucha gente y algunas empresas realmente grandes que buscan grandes audiencias. Es importante recordar lo que estás tratando de lograr al crear un anuncio. Hay dos motivaciones: crear presencia de producto y acción directa (es decir, la descarga del juego). En el caso de los juegos tenemos la posibilidad de poner anuncios en otros juegos y aplicaciones que se utilizan en el mismo dispositivo en el que queremos que los jugadores prueben nuestro juego. El camino entre el

10 www.pocketgamer.biz/events

11 indieprize.org

anuncio y la tienda de aplicaciones es completamente directo. Un clic y es posible comprar o descargar el juego. Esto es algo increíble, no hay ningún otro medio que tenga este tipo de experiencia sin fricción.

Otra de las peculiaridades a tener en cuenta es que cuanto mayor sea el alcance (rango de jugadores) que buscas, más cara será cada una de las instalaciones. Esto se debe a que la compra de espacio en una red de publicidad se basa en un proceso de licitación y los resultados se calcularán sobre la base del coste por instalación (Cost Per Install), coste por mil (Cost Per Mille) o una mezcla de ambos conocida como eCPM (CPM efectivo). Existen redes publicitarias como Chartboost.com o AppFlood.com que ofrecen publicidad cruzada.

La publicidad basada en vídeo está creciendo y permite al jugador entender al instante la naturaleza del juego que se muestra. Esto se combina a menudo, con Unity Ads[12], Vungle.com y AdColony.com con incentivos dentro del juego, como por ejemplo monedas gratis. Este tipo de incentivos es diferente de los incentivos externos como los ofrecidos por proveedores como Tapjoy y, esto es muy importante, no están permitidos por Apple.

Los eventos regulares y de comunicación con la comunidad nos permiten mantener y hacer crecer nuestra audiencia. Sobre la base de auténticas experiencias sociales, tales como la grabación de vídeos del juego y el intercambio de datos en la comunidad (puntuaciones más altas, etc.), los jugadores pueden ayudarte a llegar a sus amigos y a otros jugadores potenciales a través de Facebook, Twitter, YouTube y Everyplay.

El año 2015 marcó un punto en el que el papel del compromiso social con los juegos alcanzó niveles extremos y personalidades del YouTube como Pewdiepie y Yogscast están

12 unityads.unity3d.com

teniendo un impacto significativo en el despegue de los juegos, incluyendo los de móvil. Equipos como Hipster Whale y Serious. ly han llegado a acuerdos con estos influenciadores para ayudar a impulsar sus juegos a lo más alto de las listas de las tiendas de aplicaciones. Sin embargo, como ocurre con cualquier medio esto se está convirtiendo en algo cada vez más comercial, así que sigue siendo importante tener en cuenta la audiencia y considerar por qué un YouTuber involucrándose con tu juego va a ser interesante de cara a tus jugadores potenciales.

En el lado de los deportes electrónicos ha sido interesante ver que, además de las grandes audiencias en los canales en línea como Twitch, incluso los canales de televisión como ESPN han comenzado a llevar audiencias masivas a ver los juegos tan seriamente como ven otros deportes más físicos. El nivel de talento y profesionalismo en el mercado de los deportes electrónicos es ahora significativo y los desarrolladores de juegos están empezando a considerar cómo impactará esto al diseño del juego. Sin embargo, todavía son muy pocos los juegos móviles que legítimamente pueden decir que ha ganado una masa de seguidores lo suficientemente sólida.

Al final, a pesar de todas las diferencias en los detalles, el mundo móvil es como cualquier otra plataforma. Tenemos que conseguir, retener y rentabilizar nuestra audiencia. Eso solamente sucede si entretenemos a los jugadores de un modo que funcione bien con sus dispositivos. Los móviles son, probablemente, los dispositivos más sociales y más penetrantes en la historia de la humanidad. Los juegos para móviles están creciendo a pesar de las dificultades. Las lecciones aprendidas afectarán a todos los aspectos del desarrollo de videojuegos.



BY Alex Jonsson

Desarrollo Móvil e Internet de las Cosas

Hace una década había dos gigantes: la industria del móvil y la Internet de las Cosas (IoT, del inglés Internet of Things)[1] vivían vidas separadas sin hablarse demasiado la una a la otra. Cada una de ellas disfrutando de unos volúmenes de negocio de miles de millones de dólares, pero con poco conocimiento compartido y pocas ideas necesarias sobre cómo proporcionar en última instancia servicios de extremo a extremo para los clientes, ya que eso hubiera requerido una atención simultánea entre ambos gigantes. Si los principales actores de la internet industrial hubieran seguido el camino marcado por el mercado de consumo desde un principio y hubieran confiado más en terminales móviles como teléfonos y tablets para proporcionar servicios a los clientes finales, en este capítulo habrían florecido una gran cantidad de herramientas, plataformas y servicios disponible hoy. Sin embargo no ha sido este el caso, y solo en los últimos años ha crecido el interés en la IoT con la participación de smartphones modernos, lo que resulta en menos aunque interesantes soluciones que vale la pena difundir en estas páginas.

Uno de los hilos conductores que vemos es que varios sistemas de hardware tradicionalmente cerrados están abriendo sus APIs de servicios y moviendo bases de código a GitHub[2] y otros repositorios de código abierto con el fin de simplificar el acceso de desarrolladores a sistemas y servicios industriales.

1 también conocida como M2M (machine-to-machine, máquina a máquina), internet embebida o industrial

2 github.com

Es muy prometedor tener a grandes corporaciones industriales como Ford, General Electric, Bosch, Cisco y Siemens migrando a estándares más abiertos y exponiendo las APIs de sus productos y servicios a medida a los desarrolladores acostumbrados a las tecnologías basadas en IP. Para nosotros en la industria móvil, es fácil ver un interés creciente por conectar los teléfonos a distintas cosas, y cada vez hay más dispositivos de hardware que dependen de aplicaciones móviles de soporte para obtener su gama completa de funcionalidad. Salir del modelo autónomo y cerrado les permite lanzar productos antes y tener la capacidad de agregar una función a lo largo del tiempo mediante la actualización del firmware, en conjunción con la funcionalidad y el número de aplicaciones para móviles. Esta manera de lanzar productos ha demostrado ser muy útil también desde el punto de vista competitivo y se ha extendido desde el mundo digital al físico, gracias a la arquitectura de software dinámica y a las actualizaciones over-the-air (OTA), por no hablar del hecho de conseguir que los usuarios estén acostumbrados a usar cosas que apenas funcionan cuando se adquieren.

Nuevos Roles para el Móvil

Los primeros usos para los dispositivos móviles consistían en el equivalente a ventanas que mostraban lo que los dispositivos inteligentes estaban haciendo. Actualmente, los dispositivos y las aplicaciones móviles pueden controlar dispositivos IoT de forma remota e incluso actuar como un sensor cuando, por ejemplo, la posición GPS de un usuario es esencial para muchos tipos de localización de contenidos y servicios contextuales. En otros casos, el smartphone es una puerta de enlace o proxy para sensores, como en el caso de un dispositivo de seguimiento deportivo con conectividad bluetooth. Otras funcionalidades

habituales son el control remoto, la visualización, el almacenamiento, la funcionalidad sin conexión y la autenticación de usuarios. Visita *postscapes.com/internet-of-things-examples* para ver más ejemplos de diferentes ámbitos, como el de la salud, el de las infraestructuras urbanas, la monitorización del medioambiente o la industria tradicional.

Herramientas del Negocio

Desde el punto de vista del desarrollador, la aparición y la popularidad de la creciente gama de herramientas de terceros y kits de desarrollo muestran que existe un interés considerable en las aplicaciones de la IoT. En el momento de escribir estas líneas, la industria móvil en todo el mundo ha crecido un 13% solo el último año y todavía es más o menos una carrera de dos caballos entre Android y iOS. La parte industrial muy probablemente imitará esta relación, aunque habrá más dispositivos de Apple representados en los mercados de Estados Unidos y Europa. Muchos desarrolladores de la IoT siguen utilizando los kits de herramientas de Apple y Google para el desarrollo, aunque han aparecido muchas nuevas alternativas para desarrollar aplicaciones por otros medios y utilizando otros lenguajes. Estos son algunos ejemplos de herramientas típicas para el desarrollo dentro de la IoT:

- **IBM Mobile First**, herramientas empresariales e industriales, anteriormente conocido como Worklight, más centrado en iOS: *ibm.com/mobilefirst*
- **Ionic**, herramientas de desarrollo de interfaces, dirigidas a las interfaces de usuario multiplataforma: *ionicframework.com*
- **Xamarin**, desarrollo en C#, soporte para muchas funcionalidades nativas: *xamarin.com*

- **Evothings**, paquete de desarrollo rápido para aplicaciones nativas/híbridas utilizando JavaScript: *evothings.com/download*
- **Waygum**, una plataforma de aplicaciones para IoT móvil, con interfaces móviles: *waygum.com*

Los Detalles Diabólicos

Cada segmento de mercado de la IoT, desde los wearables a la automatización de bienes raíces y desde las aplicaciones médicas a la gestión de vigilancia y de flotas, tiene sus propios desafíos y necesidades especiales: el uso sin conectividad, los grandes volúmenes de datos, la necesidad de encriptación fuerte, la interacción en tiempo real sin retrasos o las altas demandas de ancho de banda o de accesibilidad a escala global. Ninguna herramienta o librería los satisface todos. El contexto en combinación con un conjunto equivocado de prestaciones puede limitar tus opciones de implementación. La funcionalidad adaptada al contexto, a menudo en combinación con estándares abiertos, es la manera de prevalecer en las primeras fases de este mercado, hasta que se haya alcanzado el nivel de madurez que muchos otros segmentos en el ecosistema móvil han alcanzado.

"¿Qué hace que una aplicación sea IoT?", podría preguntar alguien durante una cena. "¿No es como cualquier viejo cliente de base de datos puesto en un smartphone? ¿Dónde está la gran novedad?" Bueno, el acceso a la nube es un componente importante, pero no el único. El diablo está en los detalles, como siempre. Digamos que estás auditando el consumo de energía en tu casa y hay un gadget conectado a la caja de fusibles, el cual se comunica regularmente con un servicio a través de una interfaz RESTful vía HTTP, y por lo tanto tú

puedes acceder a una página web plagada de gráficos a través de tu tablet. Pero de repente tu cliente quiere leer los datos históricos cuando esté desconectado, o leer una etiqueta NFC, o quizás enviar un SMS de cortesía avisando cuando la sauna esté a la temperatura adecuada. ¿O por qué no escanear algunos iBeacons utilizando el sistema de bluetooth de baja energía del teléfono? De pronto, tu proyecto ha crecido mucho más allá de la seguridad del contenedor web, por no mencionar el presupuesto de tu proyecto web móvil. En este momento, una solución nativa utilizando herramientas de Apple y Google, o para muchos casos una aplicación híbrida, es el camino a seguir.

Optar por lo Híbrido para Prototipado Rápido

Como ingeniero de software, hay una buena razón para considerar herramientas híbridas para el desarrollo y para la creación de prototipos, especialmente cuando tienes un presupuesto ajustado o necesitas hacer una aplicación con muchas versiones (conjuntos de funcionalidades, idiomas, distintas interfaces de usuario, etc.). Las tecnologías web y de scripting son intrínsecamente más fáciles de asimilar para el desarrollador principiante[3]. También permiten un desarrollo más rápido de una interfaz de usuario decente, pudiendo después seleccionar de entre una variedad de librerías potentes de terceros para implementar las partes intrínsecamente nativas de tu aplicación.

Una de las tecnologías de base más populares para el tema híbrido es el proyecto Apache Cordova, hermano de su versión

3 Consulta los capítulos sobre multiplataforma y tecnologías web para más información sobre el potencial general y las limitaciones de un enfoque híbrido

comercial Phonegap[4]. Varios SDKs híbridos comerciales usan Cordova como piedra angular, por ejemplo Mobilefirst (IBM), Salesforce One (Salesforce), Evothings Studio (Evothings), Appgyver e Intel XDK. Gracias a su abierta arquitectura de plugins, tanto componentes web como componentes nativos (construidos específicamente para cada sistema operativo objetivo) pueden ser combinados con libertad, con el objetivo final de crear un proyecto de Xcode para iOS o una aplicación Android para publicar en las tiendas de aplicaciones. También existen herramientas y librerías disponibles fuera de la familia Cordova, y pese a que pocos se centran exclusivamente en el desarrollo de aplicaciones para IoT, es posible que tengan funcionalidades útiles para el desarrollo de servicios industriales.

Comunicaciones y Protocolos

Uno de los asuntos pendientes en el desarrollo para la IoT es la aparición de protocolos exóticos de comunicación para un programador móvil, con nombres como XMPP[5], MQTT[6] y CoAP[7]. Las aplicaciones para smartphones necesitan maneras para comunicarse utilizando algunos de estos protocolos para interactuar con los dispositivos de la IoT. Por suerte, ya hay algunas implementaciones disponibles, como el proyecto Eclipse Paho que incluye un cliente Android[8]. MQTT puede funcionar tanto con sockets TCP/TLS como con Websockets, lo que permite que pueda utilizarse este formato dentro de un navegador móvil.

4 phonegap.com

5 xmpp.org

6 mqtt.org

7 tools.ietf.org/html/rfc7252

8 eclipse.org/paho/clients/android

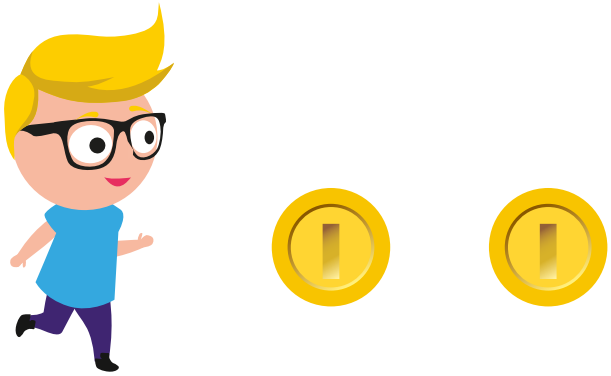
Para poder interactuar a través de conexiones de bajo nivel TCP y UDP es necesario introducir otras tecnologías como los Chromium sockets (por ejemplo los sockets Berkeley empaquetados para programadores JavaScript), disponibles como plugins para Apache Cordova.

Soportar plugins móviles para TLS (Transport Layer Security) también es un paso adelante hacia un buen nivel de seguridad de extremo a extremo, desde el sensor hasta los controles de seguridad del dispositivo IoT, salvándonos de muchas de las incertidumbres que enfrentan los servicios web y APIs expuestos al internet público. Viniendo del viejo mundo de tenerlo todo incrustarlo, para una organización que ha estado disfrutando de redes a medida sin compartir los cables con nadie puede ser aterrador obtener, de sopetón después de conectar una puerta de enlace HTTP al viejo sistema M2M, todas las posibilidades, horrores y problemas que los residentes de internet han estado viendo desde hace más de 20 años. Asegurar los mensajes y asegurar el canal es la manera de obtener paz interior.

Las aplicaciones de IoT han demostrado con el tiempo que eran más que simples clientes de bases de datos basados en la web y están haciendo mucho más que visualizar vistas de datos generadas del lado del servidor. Una segunda oleada de aplicaciones está llegando, en donde los servicios móviles de la IoT conversan directamente utilizando



radio de corto alcance y protocolos basados en IP de bajo nivel para los datos de los sensores y los mensajes de telemetría con un mínimo de sobrecarga. El estándar que prevalece aquí es Bluetooth Smart, que últimamente ha adquirido una mayor sensación de seguridad, así como las capacidades de mallado. Dos de las aplicaciones más interesantes son, por una parte, permitir que en los dos conjuntos de chips la radio bluetooth esté en modo de anuncio (broadcasting) y conecte los servicios al mismo tiempo, lo que lleva al bluetooth como un concepto más allá del concepto limitante de "un teléfono, un dispositivo". El otro es el formato abierto de bluetooth Eddystone, que permite a los usuarios consumir servicios móviles contextualmente relevantes según localización sin tener que pasar por un nodo centralizado o servidores de terceros. Permite que los usuarios finales descubran y evalúen servicios limitados a una zona geográfica, un tipo de servicio o un rol de sistema.



Aprender Más

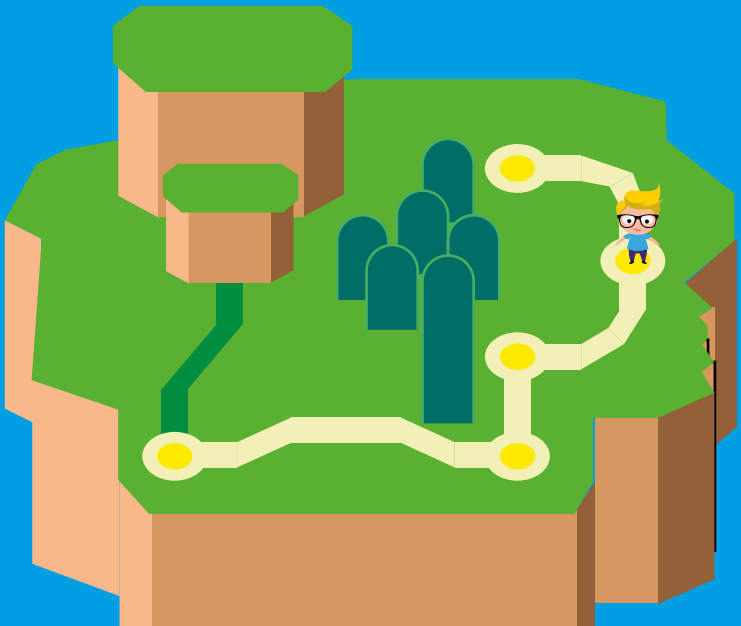
- **Artículo introductorio** comparando protocolos IoT: *electronicdesign.com/embedded/understanding-protocols-behind-internet-things*
- **A Cisco view on IoT Application Protocols**: *blogs.cisco.com/ioe/beyond-mqtt-a-cisco-view-on-iot-protocols*
- **Scaling the Internet of Things** vídeo de Yodit Stanton grabado en el ODI Summit 2015: *youtube.com/watch?v=MP2HLCNPgJ0*
- **Protocolos Eclipse IoT**: *iot.eclipse.org/protocols.html*
- **Datos en tiempo real con MQTT**, vídeo tratando temas de MQTT y de IoT: *youtube.com/watch?v=gj8mcn9oWgE*
- **Demostración de IoT utilizando WebSockets**: *developer.mbed.org/cookbook/Internet-of-Things-Demonstration*
- **Informe de Vision Mobile sobre el panorama de la IoT**: *visionmobile.com/product/commerce-of-things-2015*

Para finalizar este capítulo de una manera progresiva, a continuación se muestran algunos buenos puntos de partida, mostrando intereses de la industria, software, hardware, agregadores y proveedores de servicios.





- **Estimotes Blog**: *estimote.com*, fabricante de iBeacons y SDK móvil
- **Ejemplos y plantillas de Evothings Studio**: *evothings.com/developer*
- **IBMs IOT Foundation**: *internetofthings.ibmcloud.com*, software en la nube para IoT y aplicaciones
- **IFTTT**: *ifttt.com*, If-This-Then-That, una empresa en la nube conectando eventos a través de internet
- **Intel IoT y el Intel XDK**: *software.intel.com/en-us/iot*, herramientas de desarrollo para microcontroladores y aplicaciones móviles
- **Parse IoT**: *parse.com/products*, empresa de back-end con gran cantidad de código cliente y librerías, ejecutándose encima de Amazon
- **Phant de Sparkfun**: *data.sparkfun.com*, fabricante de hardware y accesorios de IoT, aquí enlazando a su elegante back-end de IoT, perfecto para creadores de aplicaciones que quieren controlar sus propios datos



Aplicaciones para Wearables

Después del trabajo pionero de Metawatch, Pebble y muchas más empresas, Google lanzó Android Wear en la primavera del 2014 y varios fabricantes sacaron smartwatches compatibles. Samsung sacó al mercado una gama de relojes funcionando con Tizen y Microsoft hizo lo propio con su Microsoft Band para fitness. Apple también desveló su gama de relojes Apple Watch a finales del 2014.

Los Ecosistemas

Pebble, Android Wear, Apple Watch y Samsung Tizen podrían considerarse las plataformas más importantes. También existen relojes autónomos con Android y toda una gama de dispositivos populares para monitorizar actividad de empresas como Nike, Jawbone, Fitbit, Misfit, Razer y Microsoft. Muchos wearables tienen un historial pésimo en lo que se refiere a la duración de la batería, motivo por el cual algunas empresas como Martian, Withings o Cogito aplican mejoras inteligentes a relojes tradicionales. En lugar de tener tiempos de vida de días, estos relojes aguantan seis meses o más con una sola batería.

Un hecho interesante es que las grandes plataformas de smartwatches aumentan el bloqueo alrededor de dicho ecosistema. Los smartwatches de Android requieren un dispositivo Android con certificación de Google que tenga Google Mobile Services, por lo que los dispositivos Android Open Source Platform (AOSP) no sirven. Los relojes con Samsung Tizen funcionan mejor con móviles de Samsung. Y Apple Watch,

como era de esperar, requiere un dispositivo iOS para funcionar correctamente.

De cara a la interacción, hay que diferenciar entre aplicaciones independientes que se ejecutan en el reloj y aplicaciones de acompañamiento que se ejecutan en el teléfono pero muestran contenido en el reloj. Muchos smartwatches resultan ser bastante limitados cuando se desconectan del teléfono, pero con otros es incluso posible añadir una tarjeta SIM y realizar llamadas utilizando directamente el reloj.

Si bien hay estadísticas alarmantes sobre la disminución en el uso de los wearables después de comprarlos, los wearables como categoría se han establecido firmemente. De modo que ahora que ya soportas teléfonos y tablets y posiblemente ordenadores, puedes seguir adelante y añadir otro factor a tus planes de desarrollo... y ya que estás en ello, ¡no te olvides de televisores y coches!

Smartwatches

Diseño de Experiencia de Usuario para Smartwatches

Sea cual sea la plataforma que elijas, presta atención a la experiencia de usuario de tus aplicaciones para smartwatches. Como dispones de muy poco espacio, necesitas mostrar tus intenciones muy claramente y sin información superflua. No importunes a tus usuarios con demasiadas notificaciones o exigiendo una entrada precisa de datos. Algunos dispositivos admiten interacciones táctiles, pero otros utilizan solo los botones habituales en un reloj. Los toques y otros gestos pueden ser difíciles de realizar correctamente en la pequeña superficie de un reloj, especialmente cuando el usuario está desplazándose.

Evalúa qué notificaciones pueden ser útiles para el usuario,

y si sería práctico permitirle elegir el sistema de notificación, por ejemplo vibraciones cuando esté en una reunión.

— Pebble ofrece una excelente guía sobre experiencia de usuario como parte de sus mejores prácticas de diseño[1]. La guía incluye patrones sobre navegación, diseño e interacción.
— Este artículo en profundidad de Nielsen Norman Group[2] combina un análisis del smartwatch Samsung Galaxy Gear con recomendaciones de diseño para aplicaciones para smartwatches.
— Jonathan Kohl escribió un exhaustivo artículo sobre el diseño de productos para smartwatches y wearables[3].

Android Wear

Android Wear se enfoca principalmente, tal como su nombre indica, no solo a los smartwatches sino más ampliamente a los wearables, pero en el momento de escribir este capítulo los únicos dispositivos compatibles con Android Wear son smartwatches. Fabricantes como LG, Motorola, Asus, Sony e incluso Samsung han lanzado relojes basados en Android Wear.

Tu punto de partida para desarrollar es developer.android.com/wear. Necesitarás siempre una aplicación para Android que contenga la aplicación para el wearable. Puedes escoger entre diferentes niveles de integración para soportar smartwatches:

— **Sin integración**: tus notificaciones todavía se mostrarán

1 developer.getpebble.com/guides/best-practices/design
2 nngroup.com/articles/smartwatch
3 kohl.ca/2014/lessons-learned-when-designing-products-for-smartwatches-wearables

en un smartwatch conectado. Ten presente que esto a veces provoca que el usuario se vea inundado de notificaciones, así que sé parco con ellas.

- **Notificaciones mejoradas con Android Wear**: puedes mejorar las notificaciones para su visualización e interacción en el smartwatch. Puedes añadirles páginas, visualizaciones a pantalla completa y acciones específicas del smartwatch.
- **Control de voz**: puedes registrar instrucciones de voz para interactuar con la aplicación en el smartwatch sin manos.
- **Wearable app**: puedes crear aplicaciones que se ejecuten directamente en el reloj y por lo tanto tenga acceso a sensores, etc. Esto resulta útil por ejemplo para aplicaciones de fitness para poder seguir rutas sin necesidad de llevar un teléfono al mismo tiempo (naturalmente, esto requiere un reloj con GPS). Puedes utilizar la mayoría de las APIs de Android en tu aplicación para wearables, apenas unas pocas librerías no están soportadas: los paquetes android.webkit, android.print, android.appwidget, android.app.backup y android.hardware.usb no pueden ser utilizados.

Puedes utilizar los emuladores de Android Wear para testear la aplicación, pero necesitarás un dispositivo real para pulir la experiencia de usuario. Para desarrollo en Android en general por favor consulta el capítulo Android. Puedes mantenerte al día de las novedades en Android Wear uniéndote a la comunidad de desarrolladores de Android Wear[4].



4 plus.google.com/communities/113381227473021565406

Apple Watch

La primera gama de relojes Apple Watch fue lanzada en el primer trimestre del 2015. Los Apple Watch vienen en diferentes tamaños y una gama de colores para satisfacer diferentes gustos (y presupuestos).

Puedes comenzar a desarrollar visitando *developer.apple.com/watchkit*. Si bien no se pueden crear aplicaciones independientes puras con la versión inicial de WatchKit, tienes estas opciones a tu disposición:

- **Notificaciones interactivas**: crea notificaciones que se muestren en el Apple Watch y con las que el usuario pueda interactuar.
- **Glances**: información de solo lectura.
- **Aplicaciones WatchKit**: las aplicaciones pueden contener extensiones WatchKit que se ejecutan en segundo plano en el iPhone y de manera remota actualizan e interactúan con la interfaz de usuario que se muestra en el Apple Watch.
- **ClockKit**: añade información adicional en la superficie del reloj con lo que se conoce como Complications.

Samsung Tizen

La gama inicial de smartwatches de Samsung operaba con versiones propietarias de Android. En 2014, Samsung comenzó a lanzar relojes "Gear" basados en Tizen e incluso reescribió el firmware de sus relojes existentes para utilizar Tizen. Actualmente el curvado y autónomo Samsung Gear S es probablemente uno de los smartwatches más emblemáticos que se comercializan.

Tus puntos de partida para el desarrollo para smartwatches Tizen son *developer.samsung.com/gear* y *developer.tizen.org*. Puedes empezar a dar soporte a smartwatches Tizen enviando

notificaciones interactivas[5]. La forma más fácil de desarrollar aplicaciones independientes para smartwatches Tizen es incrustando una aplicación Tizen HTML5 dentro de tu aplicación Android. Para la comunicación entre tu aplicación en el teléfono y tu aplicación Tizen tienes que utilizar el protocolo SAP SP (Samsung Accessory Protocol Service Profile, un nombre que solo podría gustarle a una madre), que básicamente consiste en un protocolo basado en matrices de bytes que requiere serialización. Para mantenerte al día con las últimas novedades sobre Samsung Gear sigue el canal de Twitter de desarrollo Samsung *@samsung_dev*.

Pebble

Pebble es, junto con Metawatch, uno de los pioneros del movimiento smartwatch. A día de hoy Pebble tiene una variedad de relojes con formas redondeadas y pantallas a color. Tu punto de partida para el desarrollo en Pebble es developer.getpebble.com. Las aplicaciones autónomas están escritas en C. También puedes usar el IDE basado en navegador cloudpebble[6] en cualquier sistema operativo, o el SDK de Pebble para sistemas Mac y Linux. También puedes utilizar JavaScript para desarrollar aplicaciones de acompañamiento que se ejecuten en el teléfono pero que puedan mostrar actualizaciones de estado y demás en el reloj. Una versión inicial aunque no oficial de emulador está disponible en GitHub[7]. Con aplicaciones en segundo plano, acceso a sensores y opciones de comunicación con AppMessage/AppSync puedes crear grandes aplicaciones de Pebble. Puedes seguir a Pebble en Twitter en *@PebbleDev*.

5 developer.samsung.com/galaxy#rich-notification

6 cloudpebble.net

7 github.com/PebbleDev/qemu_pebble

Monitores de Actividad

Hay muchos monitores de actividad que ofrecen oportunidades para desarrolladores. A menudo, la única opción es tener acceso a los datos en la nube para poder crear tus propias estadísticas, pero algunos dispositivos también son compatibles con aplicaciones independientes. Estos son los monitores más populares junto con las páginas correspondientes para desarrolladores:

- **Fitbit**, *dev.fitbit.com*
- **Garmin**, *developer.garmin.com/connect-iq*
- **Jawbone**, *jawbone.com/up/developer*
- **Misfit**, *build.misfit.com*
- **Microsoft Band**, *developer.microsoftband.com*
- **Nike**, *dev.nike.com*
- **Polar**, *developer.polar.com*
- **Razer Nabu**, *developer.razerzone.com/nabu*

Gafas Inteligentes

Las gafas inteligentes aumentan la realidad con información basada en el contexto, ya sea sobre lo que estás mirando o sobre tu tarea actual o ambas cosas.

Actualmente las gafas inteligentes se utilizan principalmente en escenarios de trabajo, en tanto que sus precios y limitado tiempo de funcionamiento son menos atractivos para los consumidores finales. Google decidió terminar la comercialización de las gafas inteligentes más conocidas, las Google Glass, y ahora están siendo remodeladas para un uso más de negocios. Las HoloLens de Microsoft fueron nombradas como uno de los mejores "inventos del año" por la revista Time en 2015.

Estas son algunas de las gafas inteligentes que están actualmente en producción:

- Epson Moverio[8]
- Google Glass[9]
- Microsoft HoloLens[10]
- Sony Smart Eyeglass[11]
- Vuzix M100[12]
- Westunitis InfoLinker[13]

8 epson.com/cgi-bin/Store/jsp/Landing/moverio_developer-program.do

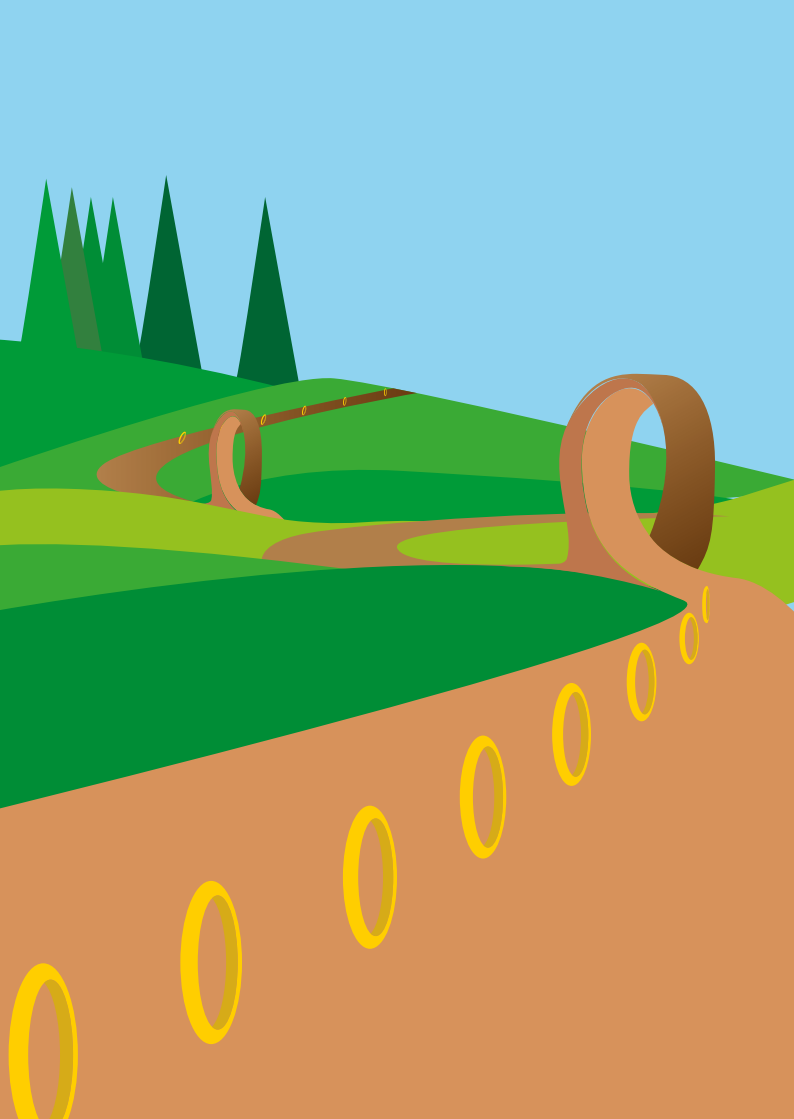
9 developers.google.com/glass/distribute/glass-at-work

10 microsoft.com/microsoft-hololens/developers

11 developer.sony.com/develop/wearables/smarteyeglass-sdk

12 vuzix.com/developer

13 westunitis.co.jp/web/global/wearable/infolinker



BY Dean Churchill

Seguridad

Los lectores de esta guía saben cómo de extendidos están los smartphones y cómo de útiles pueden ser las aplicaciones móviles. Los dispositivos móviles son además mucho más personales de lo que jamás lo han sido los ordenadores. La gente se despierta con sus teléfonos, permanecen cerca de ellos todo el día y duermen a su lado por las noches. Con el tiempo se convierten en nuestros 'colaboradores' de confianza.

Muchas de estas aplicaciones aprovechan esta cercanía y confianza. Por ejemplo, tu teléfono puede ser tratado como un eslabón en el proceso de autenticación para acceder a tu cuenta bancaria. O tu tableta puede tener acceso directo a las películas en línea que has comprado. Tu dispositivo podría incluso contener una cartera con dinero real para realizar pagos mediante tecnología NFC o con dinero virtual como bitcoins.

Las aplicaciones móviles están atrayendo la atención de hackers y ladrones cuyos intereses se extienden mucho más allá de conseguir gratis una aplicación de 99 céntimos. En el tercer trimestre del 2015 Kaspersky Lab detectó 323.374 nuevos programas móviles maliciosos[1]. Las clásicas defensas de red y de punto final (como los antivirus) no son suficientes. Incorporar medidas de seguridad en las aplicaciones móviles es un tema crítico.

La arquitectura de las aplicaciones móviles sigue evolucionando. Algunas aplicaciones son solo nativas y requieren de bases de código muy diferentes para cada sistema operativo móvil. Algunas son webviews, poco más que la URL de un sitio web envuelta en un icono. Otras son híbridas, una combinación

1 securelist.com/analysis/quarterly-malware-reports/72493/it-threat-evolution-in-q3-2015

de funcionalidad de aplicación nativa con webviews. La mayoría de aplicaciones móviles necesitan conectarse con los servicios de back-end utilizando tecnologías web para recibir o actualizar información. Como ocurre con las aplicaciones web, la seguridad para aplicaciones clásica debe ser aplicada a las aplicaciones móviles. La entrada de datos debe ser validada en lo que se refiere a tamaño, tipos y valores permitidos. El manejo de errores debe proporcionar mensajes de error útiles para los usuarios a la vez que evite el filtrado de información sensible. Las pruebas de penetración en aplicaciones son necesarias para asegurarse de que los controles de identificación, autenticación y autorización no pueden ser superados. El almacenamiento en los dispositivos debe ser inspeccionado y probado para asegurar que los datos sensibles y las claves de cifrado no se almacenan en texto plano. Los archivos de registro no deben capturar contraseñas u otra información sensible. La configuración SSL debe ser probada.

Los usuarios desean utilizar tus aplicaciones de forma segura y no quieren sorpresas desagradables. Su dispositivo móvil puede exponerlos a mayores vulnerabilidades, por ejemplo su ubicación podría ser rastreada utilizando el GPS incorporado, la cámara y el micrófono podrían ser utilizados para capturar información que preferirían mantener privada, y así sucesivamente. Las aplicaciones también se pueden crear para acceder a información confidencial como la agenda de contactos. Y aplicaciones maliciosas pueden hacer llamadas telefónicas de manera encubierta y enviar mensajes SMS a números de tarificación adicional.

El desarrollador de la aplicación puede estar preocupado por su reputación, pérdida de ingresos o perdida de la propiedad intelectual. Las empresas quieren proteger los datos empresariales a los que los usuarios pueden acceder desde sus dispositivos, posiblemente utilizando tu aplicación. ¿Es posible

mantener sus datos separados y protegidos de cualquier otra cosa que el usuario haya instalado?

Amenazas para tus Aplicaciones

En algunas plataformas (iOS y Android en particular), desactivar la verificación de firmas es una práctica bastante común. Tienes que considerar si para ti es importante o no el que alguien pueda modificar tu código y ejecutarlo en un dispositivo jailbreakeado o rooteado. Un preocupación evidente sería la eliminación de una verificación de licencia, lo que podría dar lugar a que tu aplicación sea robada y utilizada de forma gratuita. Menos evidente, pero más grave, es la amenaza de inserción de código malintencionado (malware) que pueda robar información de tus usuarios o inyectar contenido ilícito y destruir la reputación de tu marca.

La ingeniería inversa sobre tu aplicación puede dar a un hacker el acceso a una gran cantidad de datos sensibles, como las claves de cifrado para películas protegidas con DRM, el protocolo secreto para hablar con tu servidor de juegos online o la manera de acceder a dinero almacenado en el teléfono para tu sistema de pago móvil. Solamente hace falta un hacker y un teléfono jailbreakeado para explotar cualquiera de estas amenazas.

Si tu aplicación maneja dinero real o contenido valioso es necesario tomar todas las medidas posibles para protegerla de ataques Man-At-The-End (o MATE, en los que el atacante tiene acceso físico al dispositivo). Y si estás implementando un estándar DRM, tendrás que seguir las reglas de robustez que hacen obligatorias las medidas de auto-protección.

Protegiendo tu Aplicación

Escondiendo el Plano de tu Código

Algunas plataformas móviles se programan utilizando código administrado (Java o .NET), integrado por código de bytes ejecutado por una máquina virtual en lugar de hacerlo directamente en la CPU. Los formatos binarios para estas plataformas incluyen metadatos que establecen la jerarquía de clases y dan el nombre y tipo de cada clase, variable, método y parámetro. Los metadatos ayudan a la máquina virtual a implementar algunas de las características del lenguaje (por ejemplo la reflexión). Sin embargo, los metadatos también son muy útiles para un hacker que esté tratando de realizar ingeniería inversa del código. Hay programas de decompilación de libre acceso que regeneran el código fuente a partir de los códigos de bytes y hacen fácil la ingeniería inversa.

La plataforma Android tiene la opción de usar la Interfaz Nativa Java (JNI, del inglés Java Native Interface) para acceder a funciones escritas en C y compiladas en código nativo. El código nativo es mucho más difícil que sea objeto de ingeniería inversa en comparación con Java, por lo que se recomienda para cualquier parte de la aplicación donde la seguridad sea de suma importancia.

"gcc" es el compilador utilizado normalmente para generar código nativo para Android, mientras que su gemelo "clang" se utiliza para iOS. La configuración predeterminada para estos compiladores prepara cada función para ser exportada desde un objeto compartido y la agrega a la tabla de símbolos dinámicos en el binario. La tabla dinámica de símbolos es diferente de la tabla de símbolos utilizada para la depuración y es mucho más difícil de desmantelar después de la compilación. El volcado de los símbolos dinámicos puede dar a un hacker un índice muy útil de cada función en el código nativo. Utilizar correctamente

el `-f visibility` conmutador del compilador[2] es una manera fácil de hacer que sea más complicado comprender el código.

El código Objective-C compilado contiene código máquina y una gran cantidad de metadatos que pueden proveer a un atacante de información sobre nombres y la estructura de llamadas de la aplicación. Actualmente existen herramientas y scripts para leer estos metadatos y guiar a los hackers, pero no hay herramientas para ocultarlos. La manera más común de construir una interfaz gráfica para iOS es utilizando Objective-C, pero el método más seguro consiste en minimizar su uso y cambiar a C o C++ para todo lo que no sea la interfaz gráfica de usuario.

Escondiendo el Control de Flujo

Incluso si todos los nombres están ocultos, un hacker bueno todavía puede averiguar cómo funciona el software. Hay herramientas comerciales de protección de código administrado que pueden ocultar deliberadamente el camino a través del código recodificando las operaciones y rompiendo los bloques de instrucciones, lo que hace la decompilación mucho más difícil. Con una buena herramienta de protección, un intento de decompilar un binario protegido terminará o bien en un fallo del decompilador o en un código fuente no válido.

Decompilar código nativo es mucho más difícil, pero se puede conseguir. Incluso sin una herramienta, no se necesita mucha práctica para ser capaz de seguir el control de flujo en el código ensamblador generado por un compilador. Las aplicaciones con un fuerte requisito de seguridad necesitarán una herramienta de ofuscación para el código nativo, así como para el código administrado.

2 http://gcc.gnu.org/wiki/Visibility

Protegiendo las Comunicaciones de Red

Las comunicaciones de red son vulnerables a los ataques basados en fisgonear y en inyección. Las aplicaciones pueden ser instaladas e inspeccionadas en emuladores o simuladores. Los analizadores de red están disponibles gratuitamente y son capaces de monitorear, interceptar, cambiar y redirigir el tráfico de red. Algunos gobiernos vigilan las comunicaciones electrónicas para los temas censurados. Protege todas las comunicaciones utilizando HTTPS en lugar de HTTP. Las descargas de librerías JavaScript de sitios públicos deberían utilizar HTTPS, ya que los hackers usando ataques MATE pueden inyectar código malicioso en la descarga si se utiliza HTTP. Las descargas de contenido estático, como imágenes, de sitios públicos, deberían utilizar también HTTPS, ya que los hackers podrían sustituir las imágenes por contenido malicioso. Una forma de incrementar la seguridad del transporte es utilizar el cifrado asimétrico entre el servidor y la aplicación móvil (usando pares de claves públicas/privadas) para proporcionar seguridad de extremo a extremo. Las aplicaciones móviles deben validar que el nombre de servidor o el nombre común del servidor HTTPS al que se conectan es el correcto y el que se espera. Para las aplicaciones y los datos corporativos sensibles, instala servidores de red privada virtual (VPN), e instala los clientes VPN en los dispositivos móviles. Las redes VPN generalmente proporciona autenticación fuerte y un transporte seguro más allá de HTTPS.

Protegerse contra la Manipulación de Datos (Tampering)

Puedes proteger el código base aún más detectando los intentos de manipulación en la aplicación y respondiendo a esos ataques. El código criptográfico siempre debe utilizar algoritmos de cifrado estándar relativamente seguros (como AES o ECC) pero, ¿qué pasa si un atacante puede encontrar las claves de cifrado en tu archivo binario o en memoria en tiempo de ejecución? Esto podría resultar en que el atacante tenga acceso a contenido valioso. Incluso si utilizas criptografía de clave pública y solo se expone una parte del par de claves, todavía tienes que considerar lo que ocurriría si un atacante intercambiara esa clave por otra de la que él conociera la otra mitad. Necesitas una técnica para detectar cuándo el código ha sido manipulado. Hay herramientas disponibles que encriptan/desencriptan código sobre la marcha, llevan a cabo sumas de verificación contra el código para detectar su manipulación, y reaccionan cuando el código ha cambiado.

Las comunicaciones entre la aplicación móvil y los servicios de backend pueden ser monitorizadas y pirateadas. Incluso cuando se utiliza HTTPS, un proxy web interceptor (como Paros) puede ser configurado en una conexión WiFi para que inspeccione el tráfico encriptado. Los atacantes pueden entonces manipular los datos en tránsito, con fines de lucro o entretenimiento. Así que, si se están enviando datos realmente sensibles a través de HTTPS, considera encriptar/desencriptar datos en la aplicación móvil y en el servidor, de modo que los rastreadores de red solamente puedan ver datos cifrados.

Protegiendo Algoritmos Criptográficos

Una herramienta activa contra la manipulación puede ayudar a detectar o prevenir algunos ataques a claves criptográficas, pero no permitirá que las claves permanezcan siempre ocultas. La criptografía de caja blanca tiene por objeto aplicar los

algoritmos de cifrado estándar de manera que las claves permanezcan ocultas. Algunas versiones de criptografía de caja blanca utilizan complejos enfoques matemáticos para obtener los mismos resultados numéricos de tal manera que sea difícil de deducir por ingeniería inversa. Otras insertan claves en tablas de consulta y máquinas de estados que es difícil que sean objeto de ingeniería inversa. La criptografía de caja blanca sin duda te será necesaria si vas a escribir código DRM o necesitas almacenamiento de datos de alta seguridad.

Mejores Prácticas

No Almacenes Secretos o Información Privada

Minimiza la cantidad de información sensible almacenada en el dispositivo. No almacenes credenciales o claves de cifrado, a menos que se utilice un almacenamiento seguro protegido por una contraseña compleja. En su lugar, guarda tokens de autenticación que tengan tiempo de vida y funcionalidad limitados.

Los archivos de registro son útiles para diagnosticar errores en el sistema y hacer un seguimiento del uso de las aplicaciones. Pero asegúrate de no violar la privacidad de los usuarios mediante el almacenamiento de información de ubicación o registrando información personalmente identificable de los mismos. Algunos países tienen leyes que restringen la información de seguimiento que se puede recopilar, así que asegúrate de comprobar las leyes al respecto de los países en los que tu aplicación será utilizada.

No imprimas seguimientos de pila ni diagnósticos de sistema que un hacker pueda aprovechar para penetrar a mayor profundidad.

No Confíes en el Dispositivo

Cuando diseñes una aplicación, supón que el dispositivo estará en manos de un atacante tratando de abusar de la aplicación. Realiza el mismo ciclo de desarrollo de software seguro en la creación de aplicaciones móviles que el que emplearías con los servicios de backend. No confíes ni siquiera en las bases de datos que crees para tus aplicaciones móviles, un hacker puede cambiar su esquema. No confíes en el sistema operativo para obtener protección, la mayoría de esas protecciones pueden ser anuladas de manera trivial mediante un jailbreak al dispositivo. No confíes en que los almacenes nativos de claves mantendrán los datos protegidos, se puede irrumpir en algunos almacenes de claves por fuerza bruta, a menos que el usuario proteja el dispositivo con una contraseña compleja y larga.

Minimiza los Permisos

Android tiene el concepto de permisos, mientras que iOS tiene derechos. Ambos permiten el acceso de las aplicaciones a sensores tales como el GPS y contenido sensible. En Android estos permisos deben ser especificados como parte de la creación de la aplicación, en el archivo AndroidManifest.xml y se le muestran al usuario cuando opta por instalar la aplicación en su dispositivo.

Cada permiso aumenta el potencial de que tu aplicación haga cosas nefastas, y eso puede disuadir a algunos usuarios incluso de descargar la aplicación. Así que intenta minimizar el número de permisos o de características que tu aplicación necesita.

Herramientas

Protección

Un renombrado básico de código Java puede ser realizado mediante Proguard[3], una herramienta de código abierto, y Arxan's GuardIT[4].

Dos comercializadores de herramientas de protección de código gestionado (Java y .NET) son Arxan Technologies[5] y PreEmptive Solutions[6].

Los principales vendedores de herramientas de protección de código nativo y librerías de criptografía de caja blanca son Arxan and Irdeto[7].

Los principales vendedores de herramientas de escaneo de código fuente móvil seguro son Checkmarx[8] y HP[9].

Las técnicas para proteger el código Android frente a manipulación de datos están documentadas en androidcracking.blogspot.com/. EnsureIT de Arxan te permite insertar código adicional en tiempo de compilación para detectar depuradores, usar sumas de validación para detectar cambios del código en memoria y permitir al código ser desencriptado o reparado sobre la marcha.

3 www.proguard.sourceforge.net
4 arxan.com
5 arxan.com
6 preemptive.com
7 www.irdeto.com
8 checkmarx.com
9 www8.hp.com/us/en/software-solutions/mobile-app-security/index.html

Sniffing

Una herramienta web proxy estándar es Paros[10]. Una herramienta estándar para sniffing de red disponible en las plataformas habituales es Wireshark[11].

Decompilando

Visita el decompilador Hex Rays[12].

Aprender Más

A continuación, algunos recursos y referencias útiles que pueden ayudarte:

- Apple ofrece una guía general de seguridad de software[13]. También incluye múltiples enlaces a temas más detallados de su plataforma.
- Existen cursos comerciales para iOS y Android. Lancelot Institute[14] ofrece cursos de programación segura que cubren iOS y Android.
- Un probador gratuito para SSL se puede encontrar en Qualsys Labs[15].
- Extensas guías de aplicaciones seguras y herramientas de testeo gratuitas son ofrecidas por OWASP[16], incluyendo el OWASP Mobile Security Project[17].

10 sourceforge.net/projects/paros
11 sourceforge.net/projects/wireshark
12 www.hex-rays.com
13 developer.apple.com/library/mac/navigation/#section=Topics&topic=Security
14 www.lancelotinstitute.com
15 www.ssllabs.com/ssltest
16 www.owasp.org
17 www.owasp.org/index.php/OWASP_Mobile_Security_Project

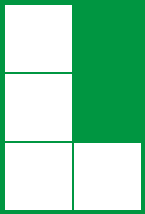
— Una opción libre para la monitorización de rendimiento para iOS y Android es la herramienta AT&T Application Resource Optimisation[18].

Conclusiones

Cada vez hay mayor confianza en las aplicaciones móviles, pero estas están expuestas a mucha gente que quisiera aprovecharse de tal confianza. El nivel adecuado de seguridad de una aplicación es algo que debe considerarse para cada caso en particular. Al final, tu aplicación va a estar ahí fuera, sola ante el peligro, y tendrá que defenderse de los piratas informáticos y otras amenazas maliciosas donde quiera que vaya.

Invierte tiempo en aprender acerca de las características y capacidades de seguridad de las plataformas móviles que tienes como objetivo. Utiliza técnicas como el modelado de amenazas para identificar los peligros potenciales relacionados con tu aplicación. Realiza revisiones de código y elimina métodos no esenciales de registro y depuración. Ejecuta una herramienta de análisis de código seguro contra el código de tu aplicación móvil para encontrar vulnerabilidades. Considera cómo un hacker podría analizar tu código y, a continuación, utiliza técnicas similares contra la aplicación en un entorno seguro para descubrir vulnerabilidades y mitigarlas antes de publicar tu aplicación.

18 developer.att.com/application-resource-optimizer



BY Sally Cain

Accesibilidad

Por Qué Es Importante la Accesibilidad

En diciembre del 2015, la Organización Mundial de la Salud (OMS) declaró en su página web que más del 15% de la población mundial tiene algún tipo de discapacidad[1] y las tasas de discapacidad están aumentando debido al envejecimiento de la población y al aumento de las condiciones de salud crónicas, entre otras causas. Esto significa que alrededor de mil millones de usuarios potenciales podrían tener dificultades para la utilización de tu aplicación en caso de que no sea accesible.

Ha habido un gran aumento en el uso de teléfonos inteligentes y tablets en la población en general, y esto incluye a las personas con discapacidad. La encuesta WebAIM Screen Reader[2] muestra que ha habido un aumento sorprendente en el uso de teléfonos inteligentes por parte de personas ciegas que usan lectores de pantalla. Las personas mayores pueden no haber utilizado un ordenador en el trabajo, pero están descubriendo que pueden llegar a acomodarse a dispositivos de pantalla táctil más rápidamente que a un teclado y un ratón tradicionales. A medida que nuestra población envejece, también lo hacen los niveles de discapacidad y esto significa que más y más personas tendrán dificultades para acceder a los servicios de la manera tradicional. Proporcionar una solución digital accesible alternativa garantizará que las personas con discapacidad puedan seguir siendo independientes. Por ejemplo, si no son capaces de salir de casa para hacer sus compras

1 www.who.int/mediacentre/factsheets/fs352/en

2 webaim.org/projects/screenreadersurvey6/

o gestionar su cuentas bancarias, la prestación de servicios en línea accesibles significará que pueden acceder a estos servicios de forma independiente. Es importante reconocer la importancia que tiene un acceso independiente a los servicios para las personas con discapacidad.

Hay muchas otras razones para hacer tu aplicación accesible:

- Implementar la accesibilidad a menudo mejora la usabilidad general. Por ejemplo, si te aseguras de que cada botón y elemento en un formulario tenga una etiqueta adecuada, esto resulta útil para todos, no solo para los que tienen discapacidades, ya que todos los usuarios sabrán cómo interactuar con él. Añadir accesibilidad en tus aplicaciones asegura una excelente experiencia de usuario para todos.
- Simplemente es un buen negocio. Por ejemplo, las personas con discapacidad tienen poder adquisitivo y si encuentran una aplicación accesible que funciona para ellos no solo la utilizarán sino que también se lo dirán a otros. Puedes descubrir un nuevo mercado significativo cuando desarrolles aplicaciones que funcionen para estos usuarios.
- El acceso a los bienes y servicios para todos viene fijado por ley en muchos países. Por ejemplo en el Reino Unido la Ley de Igualdad del 2010 exige que haya acceso a los bienes y servicios para todo el mundo y esto incluye los servicios que se proporcionan a través de un medio electrónico como los sitios web y aplicaciones. Los organismos públicos también tienen el deber de anticiparse para asegurar que sus servicios sean accesibles, para que no puedan considerar la accesibilidad como una ocurrencia tardía.
- En escenarios donde las soluciones accesibles son legalmente obligatorias, tu aplicación puede ser la única

opción para que un negocio pueda operar. Por ejemplo, tu aplicación puede ser capaz de aprovechar sectores del mercado financiados por el gobierno tales como la educación, donde legislación como la Section 508 of the Rehabilitation Act en los Estados Unidos puede exigir una solución accesible.

- La organización para la que está siendo desarrollada la aplicación puede tener un programa de responsabilidad social corporativa. Por ejemplo, la accesibilidad en webs y aplicaciones ayuda a la inclusión social de las personas con discapacidad, que es un aspecto fundamental de la responsabilidad social corporativa.
- Las plataformas móviles de Apple, Google y Microsoft aprovechan sus APIs de accesibilidad para las pruebas automatizadas de la interfaz de usuario. La creación de una aplicación accesible puede hacer tus pruebas automatizadas más fáciles.

¿Qué Funciones de Accesibilidad?

De entre tus usuarios potenciales, muchos pueden tener una discapacidad y esto puede hacer que sea más difícil para ellos usar un teléfono móvil y sus aplicaciones. Las discapacidades pueden incluir varios niveles de pérdida de visión o audición, discapacidades mentales o dificultades de aprendizaje, discapacidades físicas, problemas de destreza, y así sucesivamente.

Muchos de estos usuarios confían en software de terceros para ayudarles a utilizar su dispositivo. A este software se le suele conocer como tecnología de asistencia e incluye diversas utilidades en función del tipo de discapacidad. Tradicionalmente este tipo de software o utilidades se han tenido que 'añadir' a un dispositivo principal, a menudo a un alto precio,

con el fin de hacerlo accesible o más fácil de usar para una persona con una discapacidad. Algunos teléfonos inteligentes y tablets proporcionan ahora suficientes tecnologías de asistencia incorporadas en el sistema operativo para que algunos usuarios con discapacidad puedan utilizar los dispositivos sin necesidad de pagar por tecnología de asistencia adicional. Lo que se ofrece depende de la plataforma y la versión del sistema operativo. Sin embargo para que estas características funcionen es necesario que la aplicación se haya diseñado e implementado para soportarlas.

- **Usuarios con visión parcial** - Alguien que es parcialmente ciego se beneficia de la opción de cambiar el tamaño de fuente, el estilo de fuente y el uso de la negrita y el contraste de color. iOS, Android, BlackBerry y Windows ofrecen varias opciones para cambiar estos puntos en los ajustes. Además del clásico 'pellizca para ampliar', iOS, Android, BlackBerry y Windows ofrecen una función de ampliación o zoom que amplía una sección de la pantalla y mantiene este nivel de aumento al moverse por todo el teléfono. Esta función tiene asociados algunos gestos únicos.
- **Usuarios ciegos** - Alguien que es ciego necesita que la información de la pantalla y la navegación por la pantalla se le anuncie a través de una locución sintetizada. A esto se le llama a menudo 'lector de pantalla'. iOS fue el primer sistema operativo que ofreció un lector de pantalla incorporado llamado 'VoiceOver'. Android incluye 'Talkback' (con todas las funciones desde Android 4.1 Jelly Bean), que está ganando popularidad en la comunidad de invidentes ya que está mejorando constantemente. Windows sacó el lector de pantalla Narrator en Windows Phone 8.1, pero actualmente no está en un punto en el que se pueda utilizar para tener acceso completo al teléfono si eres un usuario ciego.

BlackBerry ofrece un lector de pantalla con funcionalidad limitada en unos pocos dispositivos.

- **Usuarios con pérdida auditiva** - Una persona con una discapacidad auditiva a menudo hará uso de un teléfono inteligente que sea compatible con audífonos y que ofrezca funciones como hace iOS con los parpadeos LED de alerta o la cancelación de ruido. También hay opciones de configuración en iOS y Android para activar los subtítulos y la ayuda en pantalla. Hacer uso de la vibración para las alertas también es útil. Algunos teléfonos también proporcionan soporte para audífonos y dispositivos TTY. Un dispositivo TTY permite a las personas con pérdida auditiva o con limitaciones del habla escribir mensajes a cualquier otra persona que tenga un TTY, usando una línea telefónica.
- **Usuarios con discapacidades físicas** - Si un usuario tiene un impedimento motor es posible que esté usando un producto de hardware para acceder al teléfono, como por ejemplo un interruptor en tanto que algunos dispositivos lo soportan. Alternativamente, podría estar haciendo uso del reconocimiento de voz para acceder al dispositivo.
- **Usuarios con dificultades de aprendizaje** - Si un usuario tiene un deterioro cognitivo o dificultad de aprendizaje, dependiendo de en qué consista la discapacidad puede hacer uso de la configuración que utilizaría un usuario con deficiencia visual. Especialmente aspectos como las opciones de color. Otros usuarios pueden hacer un mayor uso del reconocimiento de voz.

Las personas con discapacidad ven su experiencia global afectada por lo bien que funciona una aplicación con la tecnología de asistencia. A medida que estas características están integradas en el sistema operativo y se pueden activar en la configuración, es

importante que como desarrollador tengas en cuenta que se puedan usar con tu aplicación y te asegures de testearlo.

Como los lectores de pantalla y la ampliación de pantalla en el sistema operativo hacen uso de sus propios gestos, los gestos en tu aplicación pueden verse afectados cuando los lectores de pantalla o la ampliación están habilitados. Por ejemplo, un usuario de lector de pantalla puede navegar por una pantalla deslizando el dedo a izquierda y derecha o explorar la pantalla moviendo su dedo por la superficie del dispositivo en un movimiento constante. Según va deslizando el dedo y se encuentran algo debajo, se anuncia un elemento. Y ese elemento se puede seleccionar pulsando una vez y abrir tocando dos veces. Al utilizar la magnificación de pantalla, dependiendo del sistema operativo, es posible que necesite usar un gesto de tres dedos. Incluir las pruebas con funciones de accesibilidad desde el principio asegura que estos gestos son soportados por la aplicación y que cualquier rediseño puede realizarse sin impacto para los usuarios.

Una de las mejores maneras de aprender más acerca de estas características es activarlas y probarlas por ti mismo en diferentes aplicaciones.

Directrices en el Diseño de Aplicaciones

Las APIs de accesibilidad buscan textos en unos atributos específicos de elementos estándares de la interfaz de usuario. Los lectores de pantalla utilizados por personas invidentes, como VoiceOver y TalkBack, transforman estos textos en voz sintética que el usuario escucha. El software lector de pantalla también puede determinar el tipo de control y atributos relacionados para facilitarle al usuario más información contextual, sobre todo si no hay texto disponible. Es importante que el usuario entienda qué es la etiqueta del control, qué es el control y cómo interactuar con él. En algunos casos

puede haber otras etiquetas secundarias para aportar información adicional.

Del mismo modo que los desarrolladores web hacen uso de estándares y directrices tales como las WCAG 2.0 para hacer páginas web accesibles, es importante que los desarrolladores de aplicaciones hagan lo mismo. Actualmente no existe un estándar de facto de la industria para accesibilidad en aplicaciones, aunque sí que existen varios estándares que pueden ser de utilidad.

El estándar internacional, ISO 9241-171 ('The Ergonomics of Human-system Interaction: Guidance on Software Accessibility')[3] es un estándar útil, ya que es independiente de la plataforma. Cubre los elementos de accesibilidad y usabilidad para una amplia gama de software.

El Royal National Institute of Blind People (RNIB)[4] ha creado un estándar de aplicaciones y proceso de pruebas para todo tipo de discapacidades basado en su experiencia en este ámbito. Su estándar para aplicaciones nativas refleja también los principios de la norma ISO 9241-171. También ofrecen asesoría y formación a organizaciones y agencias en esta área y gestionan una acreditación que se puede conceder a las aplicaciones que se consideran accesibles tras un proceso de auditoría.

La BBC ha desarrollado un conjunto de Directrices de Accesibilidad para móviles[5] que utilizan internamente para su contenido móvil. Su guía cubre sitios web móviles y aplicaciones nativas y híbridas. Afirman que "han de considerarse como un estándar para los empleados y los proveedores de la BBC, aunque también pueden ser referencia para cualquier persona involucrada en desarrollo móvil".

A continuación se exponen algunos de los principios a tener en

3 www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39080

4 rnib-business.org.uk

5 www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile_access.shtml

cuenta a la hora de desarrollar una aplicación. Si te adhieres a ellos, también le darás a tu aplicación las mejores opciones para interactuar con la tecnología de asistencia que el usuario pueda estar ejecutando en conjunción con tu software:

APIs y Directrices para Interfaces de Usuario

- Averigua qué características de accesibilidad y APIs tiene tu plataforma y sigue las mejores prácticas en el aprovechamiento de esas APIs si existen.
- Usa elementos estándares de interfaz de usuario en lugar de elementos personalizados cuando sea posible. Esto asegurará que si tu plataforma cuenta con una infraestructura de accesibilidad, o adquiere una en el futuro, probablemente tu aplicación se mostrará como accesible para tus usuarios.
- Utiliza la API de accesibilidad para tu plataforma, si la hay. Esto te permitirá hacer los elementos de interfaz de usuario personalizados más accesibles y significará menos trabajo por tu parte.
- Sigue las directrices de interfaz de usuario estándares en tu plataforma. Esto mejora la consistencia y puede significar un diseño más accesible por defecto.
- El usuario debe ser capaz de aplicar los ajustes de preferencias que el sistema operativo proporciona, tales como la configuración de accesibilidad.

Navegación

- La navegación debe ser lógica y coherente. Por ejemplo, si en cada pantalla existe un botón para ir atrás debe estar ubicado en el mismo lugar en todas las pantallas y ser etiquetado de forma consistente.

— Permite la navegación programática de tu interfaz de usuario. Esto no solo permitirá que tus aplicaciones se utilicen con un teclado externo, sino que también mejorará la accesibilidad de tu aplicación en plataformas como Android en la que la navegación se puede realizar con un trackball o un d-pad virtual.

Elementos de Usuario

— Todos los elementos de usuario deben ser visibles y operables a través de la tecnología de asistencia, a menos que sea obvio que no son necesarios.
— Cuando un elemento de usuario tiene un estado asociado con él, dicho estado también debe estar disponible para ser leído por tecnología de asistencia. Por ejemplo, si un botón de activación está en posición 'on', el lector de pantalla debe poder informar de ello. Los cambios de estado también deberán ser anunciados.
— Asegura que los elementos a tocar en una pantalla táctil sean de un tamaño razonable para que todo el mundo pueda seleccionarlos fácilmente.

Etiquetado

— Todos los elementos (incluyendo elementos de formulario, botones, iconos y así sucesivamente) deben estar etiquetados visualmente y mediante programación con un nombre corto y descriptivo. La etiqueta debe ser adyacente al elemento con el que se relaciona.
— Cada pantalla debe tener un nombre descriptivo único que se refiera a su contenido y ayude a la navegación.

Colores y Fuentes

- Asegura que haya un buen contraste entre los colores de fondo y de primer plano. En particular, ten en cuenta los botones que incluyen texto. ¿El contraste entre el texto y el color de fondo cumple los requisitos de los estándares WCAG 2.0 o ISO 9241-171?
- Evita utilizar el color como el único modo de diferenciar una acción. Un usuario daltónico no será capaz de identificar los errores si se le muestran en rojo los datos a corregir, por ejemplo.
- Considera el tamaño de la fuente más pequeña. ¿Es razonable considerar que la mayoría de la gente podría leerla sin dificultad?

Notificaciones

- Los mensajes de error, las notificaciones y las alertas deben ser identificables y claros. Deberían ser anunciados por los lectores de pantalla y resultar claramente visibles, teniendo en cuenta que no deben desaparecer de la pantalla demasiado pronto.
- Asegura que los mensajes de error, las notificaciones y las alertas no se transmiten solo mediante un color, un sonido o una señal táctil. Por ejemplo, una persona con pérdida auditiva no reconocerá notificaciones que solo son audibles.

Testeo

- No te olvides de probar la aplicación en el dispositivo de destino con la tecnología de asistencia integrada en el sistema operativo utilizando algo más que la última versión del sistema operativo. Cuando se prueba en un dispositivo Android recuerda que a menos que el usuario tenga un dispositivo Android puro, como un Google Nexus, es poco probable que obtenga acceso a todas las actualizaciones más recientes del sistema operativo. Esto se debe a que para las actualizaciones del sistema operativo se está a merced del fabricante del teléfono, por lo que las versiones del sistema operativo Android ahí fuera pueden ser muy diversas. A veces puede haber interferencias en la forma en que las funciones de accesibilidad deben trabajar debido a que algunos fabricantes de teléfonos añaden una capa al sistema operativo. Por lo tanto siempre se recomienda que las pruebas de accesibilidad en Android se lleven a cabo en un dispositivo Google Nexus. De esta manera puedes estar seguro de que no hay nada que interfiera con la forma en que las características de accesibilidad deben trabajar y que estás trabajando con un denominador común.
- ¡Asegúrate de que entre tus usuarios probadores haya también personas con discapacidad!

Apple, Google y Microsoft han aumentado la importancia de sus respectivos soportes de accesibilidad haciendo que el uso de la interfaz de accesibilidad sirva para mejorar sus frameworks de automatización de pruebas de la interfaz gráfica de usuario. Esto supone otro incentivo más para que los desarrolladores consideren el diseño de sus aplicaciones para que sean más accesibles.

Mirando más detalladamente las diferentes plataformas móviles se hace evidente que difieren en gran medida con respecto a sus APIs, pero están comenzando a implementar muchas de las mismas características de accesibilidad.

Controles y Elementos Personalizados

Si estás utilizando elementos de interfaz de usuario personalizados en tu aplicación, las plataformas que tienen una API de accesibilidad te permiten hacer accesibles tus controles personalizados. Para ello, tienes que exponer el control a la tecnología de asistencia que se ejecuta en el dispositivo para que esta pueda consultar las propiedades del control y mostrarlo accesible.

Puedes obtener más información sobre este proceso en Android en la presentación de Google I/O 2015[6] y la presentación de Google I/O 2013[7].

El programa para desarrolladores de Apple también dispone de recursos útiles. Echa un vistazo a sus presentaciones en vídeo tratando temas de accesibilidad en las conferencias WWDC 2014 y 2015 disponibles en el iOS Developer Center[8].

Accesibilidad en Aplicaciones Android

La última gran versión de Android, la versión 6 (Marshmallow), ha seguido mejorando el soporte a la accesibilidad mediante una nueva API de asistencia por voz creada aprovechando el trabajo previo del lector de pantalla Talkback. La accesibilidad

6 youtube.com/watch?v=euEsfNR5Zw4

7 youtube.com/watch?v=ld7kZRpMGb8

8 developer.apple.com/wwdc/videos

comenzó a ser una propuesta real con Android 4.1 (Jellybean) y se ha mejorado mucho desde entonces.

Algunas de las funciones de accesibilidad en Android incluyen cosas como las siguientes:

- **Voice Assistant (Talkback)** - Salida de voz para usuarios ciegos.
- **Tamaño de fuente** - Para usuarios con visión parcial y algunos usuarios con dificultades de aprendizaje.
- **Gestos de ampliación** - Ampliación de tipo zoom para usuarios con deficiencia visual.
- **Colores negativos** - Para usuarios con deficiencia visual y algunos usuarios con problemas de aprendizaje que prefieren una paleta de colores invertida.
- **Ajustes de color** - Para usuarios con preferencias de color específicas.
- **Detector de sonido y alertas con flash** - Para aquellos con deficiencia auditiva.
- **Subtítulos** - Se proveen títulos o subtítulos para las personas con pérdida de audición.
- **Ajuste del auricular y audio mono** - Para aquellos con pérdida de audición que utilizan auriculares.
- **Interruptor universal** - Para las personas con discapacidades físicas que prefieren acceder a las aplicaciones usando un dispositivo de hardware.
- **Menú asistente** - Para que aquellos con problemas de destreza puedan cambiar la velocidad y el tamaño del cursor o touchpad.
- **Retardo al tocar y presionar** - Para usuarios con problemas motrices.

Encontrarás algunos recursos útiles en la Support Library[9] que también incluye maneras de mejorar la accesibilidad de las vistas personalizadas.

Para obtener información específica sobre cómo utilizar la API de accesibilidad de Android, junto con mejores prácticas en accesibilidad en Android, consulta el documento de Google titulado Making Applications Accessible[10].

También puedes encontrar más ejemplos en la sección de aprendizaje de la documentación para desarrolladores, en una sección titulada Implementing Accessibility[11]. Testear la accesibilidad también está tratado en línea[12].

Accesibilidad en Aplicaciones BlackBerry

Actualmente, el sistema operativo BlackBerry tiene algunas características de accesibilidad que son útiles para personas con diversas discapacidades. Las funcionalidades que ofrece son más limitadas, aunque sí que proporcionan soporte para TTY para personas con pérdida de audición. Como quiera que solo hay un lector de pantalla disponible para algunos de los dispositivos BlackBerry y no está tan desarrollado como VoiceOver para iOS o TalkBack para Android, los dispositivos BlackBerry están siendo tomados en cuenta por muy pocas personas invidentes a día de hoy. BlackBerry ha comenzado a moverse hacia el espacio Android con 'PRIV'. Dirígete a la sección Android para información sobre características de accesibilidad.

9 developer.android.com/tools/support-library/index.html

10 developer.android.com/guide/topics/ui/accessibility/apps.html

11 developer.android.com/training/accessibility/index.html

12 developer.android.com/tools/testing/testing_accessibility.html

El BlackBerry Screen Reader[13] solamente está disponible para un número muy limitado de dispositivos BlackBerry Curve. Viene preinstalado en los dispositivos BlackBerry 10 y se puede descargar para otros dispositivos soportados. Puedes ver cuáles se soportan actualmente en el sitio web sobre accesibilidad de BlackBerry[14].

BlackBerry 10 ofrece varias configuraciones de accesibilidad para permitir a los usuarios personalizar su dispositivo. Estos incluyen, entre otros:

- **Lector de pantalla** - Convierte texto en voz sintética para usuarios con poca o ninguna visión. Pero solo está disponible en el BlackBerry Z30.
- **Modo ampliar** - Permite al usuario aumentar y disminuir la ampliación de texto y otros elementos en la pantalla.
- **Subtítulos** - Son útiles en los vídeos para las personas con pérdida de audición.
- **Configuración de pantalla** - Dan al usuario la oportunidad de cambiar el texto y los colores en la pantalla. Es útil para las personas con dificultades de aprendizaje y usuarios con visión parcial.
- **Configuración de TTY** - Para usuarios con pérdida auditiva que quieren usar usar texto en lugar de voz para comunicarse.
- **Soporte a la escucha** - Disponible en algunos teléfonos.

Encontrarás documentación sobre la creación de aplicaciones

13 mobileapps.blackberry.com/devicesoftware/entry.do?code=bsr

14 us.blackberry.com/legal/accessibility.html

accesibles para BlackBerry 10 en un artículo dedicado en el sitio web de BlackBerry.[15]

Si estás diseñando para BlackBerry 10 también hay recursos de desarrollo que incluyen algunas directrices de diseño[16].

Accesibilidad en Aplicaciones iOS

Apple fue la primera empresa que integró funciones de accesibilidad directamente en el sistema operativo. Debido a esto el soporte a la accesibilidad en iOS es un poco mejor que en Android, aunque Android se está acercando rápidamente.

Algunas de las características de accesibilidad en iOS incluyen, entre otras:

- **VoiceOver** - Lector de pantalla. Indica los objetos y textos que hay en la pantalla, habilitando tu aplicación para poder ser utilizada por personas invidentes.
- **Zoom** - Amplía el contenido de la pantalla.
- **Invertir colores** - Invierte los colores mostrados, ayudando a personas que necesitan contraste entre blancos y negros pero para los que una pantalla blanca es demasiado brillante.
- **Texto grande y en negrita** - Ayuda a un amplio rango de personas, desde los que usan gafas a aquellos con ceguera parcial, pasando por los que tengas dificultades de aprendizaje.
- **Aumentar contraste** - Mejora el contraste entre el fondo y el contenido.

15 developer.blackberry.com/native/documentation/cascades/best_practices/accessibility/accessibility_features_best_practices.html

16 developer.blackberry.com/devzone/design/bb10/accessibility.html

- **Etiquetas y subtítulos** - Para usuarios con dificultades auditivas.
- **Alertas audibles, visibles y táctiles** - Permiten elegir qué tipo de alerta utilizar para las notificaciones.
- **Control por voz y Siri** - Permite llamar por teléfono y usar ciertas funciones del teléfono mediante comandos de voz. Puede ser útil para muchos usuarios, por ejemplo aquellos con dificultades de motricidad, de aprendizaje, o de visión.
- **Soporte a audífonos** - Para personas con pérdida de audición.
- **Control con interruptor** - Para aquellos con discapacidades físicas que quieran utilizar hardware de terceros para usar la aplicación.
- **Acceso guiado** - Útil en el ámbito educativo o cuando alguien quiere limitar qué es accesible desde la pantalla para un usuario.

Si estás trabajando en iOS, asegúrate de seguir las directrices sobre accesibilidad de Apple[17]. Aportan información detallada sobre la API y son una excelente fuente de recomendaciones para maximizar la experiencia de usuario en tus aplicaciones.

Apple también ofrece ayuda sobre cómo testear la accesibilidad de tu aplicación con Voiceover[18].

17 developer.apple.com/library/ios/documentation/UserExperience/Conceptual/iPhoneAccessibility/Introduction/Introduction.html

18 developer.apple.com/library/ios/technotes/TestingAccessibilityOfiOSApps/TestAccessibilityonYourDevicewithVoiceOver/TestAccessibilityonYourDevicewithVoiceOver.html

Accesibilidad en Aplicaciones Windows

Es justo decir que Microsoft ha estado poniéndose al día con iOS y Android en cuanto a accesibilidad. Desplegaron un buen soporte para la ampliación, el agrandamiento de texto y el cambio de colores, y en Windows Phone 8.1 las cosas siguieron adelante. En Windows Phone 8.1 existe un lector de pantalla llamado Narrator, que lee el texto en voz sintética y, al igual que otros lectores de pantalla del teléfono, hace uso de sus propios gestos específicos. Está diseñado para usuarios con poca o ninguna visión, pero todavía necesita mejorar para ser tan útil como sus contrapartidas en iOS y Android. Solo puede utilizarse con algunas funciones básicas y para la navegación y no tiene tantas funciones como otros lectores de pantalla. ¡Sin embargo, es sin duda prometedor!

Algunas de las características de accesibilidad en Windows Phone 8.1 incluyen, entre otras:

- **Narrator** - Lector de pantalla todavía en beta, con funcionalidad limitada y solo disponible para la versión 8.1.
- **Tamaño de texto** - El tamaño del texto se puede ampliar para ayudar a las personas con dificultades visuales o de aprendizaje.
- **Modo alto contraste** - Este modo cambia el texto a blanco y negro y proporciona un fondo de texto sólido en vez de a base de imágenes. Es útil para usuarios con deficiencia visual y/o dificultades de aprendizaje.

- **Lupa de pantalla** - Para las personas con deficiencia visual que deseen ampliar el texto en la pantalla y cambiar el nivel de zoom. Tiene sus propios gestos.
- **Control por voz** - El usuario puede hacer llamadas, buscar en internet, abrir aplicaciones o escuchar mensajes de texto mediante control por voz. Útil para una amplia gama de personas en diferentes situaciones, incluyendo aquellas con deficiencias motrices, problemas de aprendizaje o discapacidad visual.
- **Personalizar etiquetas de navegador** - Es posible cambiar el tamaño de fuente y el color y la transparencia del fondo de los subtítulos en Internet Explorer y en las aplicaciones que hacen uso del navegador. Útil para personas con pérdida auditiva o de visión.
- **Soporte a TTY** - Permite a las personas con pérdida auditiva o del habla escribir mensajes a cualquier otra persona que tiene un TTY utilizando una línea de teléfono.
- **Cortana** - Es el 'asistente personal' disponible solo en Windows Phone 8.1. Es una característica interesante para todos los usuarios, pero será de ayuda para las personas con discapacidades, ya que se puede activar con el habla.



Descubre más sobre accesibilidad en Windows Phone[19] incluyendo Narrator y otras características.

La documentación "Accessibility for Windows Runtime Apps"[20] proporciona ayuda, ya estés desarrollando en C#/VB/C++ y XAML o en JavaScript y HTML.

Microsoft tiene sus Directrices para el Diseño de Aplicaciones Accesibles[21] que es un documento realmente muy útil. Informa sobre la API correspondiente y, en el caso que tengas que utilizar controles personalizados en XAML o HTML, proporciona ayuda sobre cómo hacerlos de una manera accesible. También recoge algunas otras guías de terceros que pueden ser útiles. Por ejemplo, si estás desarrollando en HTML será importante considerar el uso de Accessible Rich Internet Applications 1.0 (WAI-ARIA)[22], útil para hacer el contenido dinámico más accesible a los lectores de pantalla.

Una vez que hayas testeado la accesibilidad de tu aplicación[23], Microsoft te permitirá declarar que tu aplicación es accesible[24] en la tienda de aplicaciones de Windows, permitiendo así que sea descubierta por aquellos que añadan el criterio de accesibilidad en sus búsquedas.

19 windowsphone.com/en-gb/how-to/wp8/settings-and-personalization/accessibility-on-my-phone

20 msdn.microsoft.com/en-us/library/windows/apps/xaml/dn263101.aspx

21 msdn.microsoft.com/en-us/library/windows/apps/hh700407.aspx

22 w3.org/TR/wai-aria/markup

23 msdn.microsoft.com/en-us/library/windows/apps/xaml/hh994937.aspx

24 msdn.microsoft.com/en-us/library/windows/apps/xaml/jj161016.aspx

Accesibilidad en Aplicaciones Móviles Web

Como se ha mencionado anteriormente en este capítulo, se ha escrito mucho acerca de la accesibilidad web, pero se ha escrito menos sobre la accesibilidad en relación con las aplicaciones. Esto también se aplica a la accesibilidad de los sitios web móviles o de las aplicaciones móviles web (webapps). Se trata de un área que tiene un interés creciente y el World Wide Web Consortium (W3C) ha creado el 'Web and Mobile Interest Group' para discutirla e identificar qué tareas deben llevarse a cabo. El número de grupos de trabajo en el W3C que se relacionan con esta área está creciendo y pueden proporcionar ayuda y documentación útil.

El W3C ha publicado un análisis sobre el estado de la cuestión y hoja de ruta que enumera los Estándares para Aplicaciones Web en Móviles[25].

El W3C ha sugerido que todo aquello que utilice HTML y esté basado en web debería atenerse a las Web Content Accessibility Guidelines (WCAG) 2.0 y a las Mobile Web Best Practices (MWBP). Así que estas directrices son un buen lugar para empezar si eres un desarrollador de contenidos web. También encontrarás que las Relationship between Mobile Web Best Practices (MWBP) and Web Content Accessibility Guidelines (WCAG)[26] son un recurso útil.

Si tu aplicación desea imitar el aspecto y usabilidad de una aplicación nativa, debes seguir las directrices mencionadas anteriormente en este capítulo.

Como el HTML5 cada día está más soportado en las diferentes plataformas móviles, considera leer la guía de mejores

25 w3.org/Mobile/mobile-web-app-state

26 w3.org/TR/mwbp-wcag/

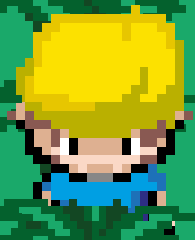
prácticas Mobile Web Application Best Practices[27], ya que es probable que se utilicen para conformar las bases de cualquier norma futura de accesibilidad de aplicaciones web móviles. Otra área clave para orientarse es Accessible Rich Internet Applications 1.0 (WAI-ARIA)[28], ya que ha sido diseñada para asegurar que las funciones de HTML más dinámicas sean accesibles para los lectores de pantalla.

Un área interesante de trabajo en el W3C es el grupo Independent User Interface (IndieUI) Working Group[29]. El grupo afirma que "la interfaz de usuario independiente (IndieUI) es una forma de que las acciones del usuario sean comunicadas a las aplicaciones web y hará que sea más fácil para las aplicaciones web trabajar en una amplia gama de contextos: diferentes dispositivos, diferentes tecnologías de asistencia (AT), diferentes necesidades de los usuarios". Esta línea de trabajo va a ser muy importante para la accesibilidad y la independencia de los dispositivos. Merece la pena echar un vistazo a la documentación que tienen disponible actualmente.

27 w3.org/TR/mwabp

28 w3.org/TR/wai-aria

29 w3.org/TR/indie-ui-context





Testeo

Hay muchos mundos paralelos en el universo de las aplicaciones móviles. Al mismo tiempo los dispositivos móviles evolucionan muy rápidamente en comparación a otros tipos de bienes. El testeo de aplicaciones móviles tiene que mantenerse a la vanguardia aún cuando el ritmo de la evolución continúa acelerándose.

El destino de las aplicaciones móviles depende de que los usuarios, al igual que las multitudes en los anfiteatros de la época romana, a menudo toman la decisión final de si una aplicación vive para luchar otro día o muere. Del mismo modo, las aplicaciones comunes y corrientes son propensas a languidecer como un mero dato estadístico más en la tienda de aplicaciones, anulando buena parte del duro trabajo invertido en idearlas, producirlas y lanzarlas. Además, el estigma de una mala calificación tiene una vida media larga y es difícil sobreponerse a él.

Testear podría ser visto como una inconveniencia, pero los fallos en tu aplicación pueden ser demasiado notorios. Y recuperar tu credibilidad es difícil cuando tu aplicación tiene una mala puntuación en la tienda de aplicaciones. Así que puedes esperar a que los usuarios decidan su destino o testear tus aplicaciones móviles para inclinar la balanza a tu favor. Tienes la oportunidad de ayudarte a ti mismo y a tu equipo de pruebas para testear de manera más efectiva tu aplicación.

Cuidado con las Especificaciones

Plataformas, redes, dispositivos, e incluso firmware... todo son especificaciones. Cualquiera de ellas podría causar problemas en tus aplicaciones. Hay varias maneras de identificar sus efectos, por ejemplo un probador puede notar diferencias en el rendimiento de la aplicación y el comportamiento de la interfaz de usuario durante la prueba con diferentes dispositivos. QuizUp utilizó pruebas automatizadas que les ayudaron a encontrar cinco problemas importantes en su aplicación para Android, provocados por las diferencias entre dispositivos, y un error específico a Android 4.0.4. Las pruebas automatizadas se ejecutaron en 30 dispositivos durante 30 minutos, lo que hizo esta prueba práctica y útil, en lugar de tener que pasar 60 horas tratando de hacer pruebas manuales de la aplicación en 30 dispositivos[1]. Necesitas conocer estas especificaciones para poder decidir si se deben abordar las diferencias no deseadas mediante modificaciones en la aplicación antes de su lanzamiento.

Por el contrario, las analíticas móviles puede ayudarte a identificar diferencias en diversos aspectos, incluyendo el rendimiento y el consumo de energía cuando la aplicación está siendo utilizada por muchos usuarios en gran variedad dispositivos. Puedes ver algunos ejemplos de diferencias en el comportamiento y en cómo se abordaron los problemas en un artículo publicado por científicos de computación de la Universidad de Wisconsin[2]. También hay disponible un libro de

1 blog.testmunk.com/quizup-mastering-android-device-fragmentation-automated-testing/

2 "Capturing Mobile Experience in the Wild: A Tale of Two Apps", disponible para descargar en static.googleusercontent.com/media/research.google.com/en//pubs/archive/41590.pdf

HP Enterprises sobre la confluencia entre las analíticas móviles y el testeo de aplicaciones móviles. Los detalles se pueden encontrar en *themobileanalyticsplaybook.com* (escrito entre otros por Julian Harty, uno de los autores de este capítulo).

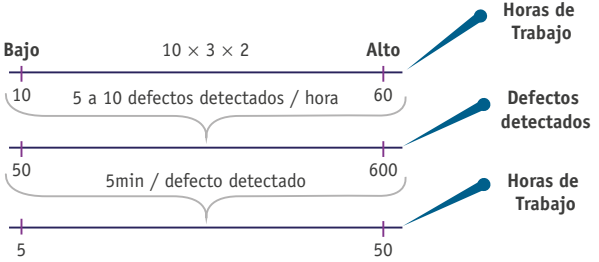
Este capítulo trata los temas generales; las pruebas para plataformas específicas se tratan en sus capítulos correspondientes.

El Testeo Necesita Tiempo - Necesitas una Estrategia

La estrategia define la cantidad de tiempo de testeo que se invierte en las diferentes partes de la aplicación móvil y durante las diferentes fases de desarrollo. Aparecen disyuntivas sobre la mejor manera de invertir el tiempo del que se disponga. Por ejemplo, probar algunas características más detalladamente versus probar en una variedad más amplia de dispositivos versus probar varios aspectos de calidad como el rendimiento, la facilidad de uso y la seguridad. Las condiciones en las que una aplicación móvil tiene que funcionar son muy amplias y considerarlas todas en las pruebas es todo un reto. Puede haber formas más productivas de obtener alguna información, por ejemplo a través del feedback de los usuarios finales y de analíticas móviles. Sin embargo, los riesgos de diferir la recopilación de información (en comparación con las pruebas internas) deben tenerse muy en cuenta. Una estrategia eficaz de testeo tiene como objetivo equilibrar ambos enfoques. Tomando en consideración el análisis de riesgos, las perspectivas de calidad y el tiempo disponible se puede crear el plan de pruebas.

Ejemplo de Cálculo

Asumamos que tu aplicación tiene diez epics (también llamados historias de usuario en este contexto) y cada epic necesita ser testeado utilizando entre uno y tres enfoques (funcionalidad, usabilidad y escenarios de usuario) a un ratio de un test cada hora. Si testeas cada epic con una perspectiva y un caso de test, te llevará diez horas testear la aplicación. Por otra parte, si testeas cada epic utilizando las tres perspectivas con dos casos de test, el testeo te llevará 60 horas. Habitualmente hacer pruebas consigue detectar defectos y otros aspectos que vale la pena reportar, lo que requiere más tiempo. Eso a su vez requiere invertir tiempo resolviendo algunas de las cosas encontradas, por ejemplo resolver un bug o realizar una mejora en la aplicación. Asumamos que cada prueba ejecutada encuentra entre 5 y 10 problemas. El tiempo necesario para procesar los defectos está entre 5 y 50 horas de trabajo. Así que en el mejor de los casos supone unas 15 horas probar la aplicación (solo el funcionamiento) y en el peor puede suponer hasta 110 horas probar la aplicación con los tres enfoques posibles.



Es posible que también necesites añadir al cálculo mucho más tiempo para probar la aplicación en una variedad de dispositivos compatibles, particularmente para aplicaciones web y aplicaciones nativas Android.

Testeo Continuo

Los lanzamientos continuos necesitan pruebas continuas. Una aplicación viable debe actualizarse de manera continua. Las actualizaciones pueden incluir correcciones para nuevas versiones de la plataforma o modelos de dispositivos, nuevas funcionalidades y otras mejoras. Por lo tanto, el testeo no es una tarea puntual; a las aplicaciones de alta calidad les conviene tener procesos de testeo continuados y optimizados, incluyendo testeo en producción. Las pruebas de testeo en producción incluyen las de validación y también la detección temprana de problemas potenciales antes de que se disparen en número.

Gestionar tu Tiempo de Pruebas

El testeo, como ya has visto, puede tomar muchas horas, muchas más de las que desearías, especialmente si te aproximas a una fecha límite como la de lanzamiento. Hay varias formas de administrar el tiempo invertido en las pruebas. Las pruebas en paralelo pueden ser más interesantes, gratificantes y productivas.

- **Reducir el tiempo de configuración**: encuentra maneras de desplegar aplicaciones de forma rápida y eficiente. Implementa mecanismos para proporcionar los datos y la configuración de las pruebas correspondientes, tanto en el dispositivo móvil como en los servidores pertinentes. Trata de tener dispositivos y sistemas 'listos para testear'.
- **Reducir el tiempo necesario para la presentación de informes y el análisis de errores**: los datos, capturas de pantalla e incluso vídeos pueden ayudar a investigar los bugs más fácil y rápidamente. Los datos pueden incluir los logs, las configuraciones del sistema, el tráfico de red y la información de tiempo de ejecución. Algunas herramientas

comerciales como HP Sprinter[3] pueden grabar acciones y capturas de pantalla para reducir el tiempo y esfuerzo necesarios para reportar y reproducir problemas.

— **Análisis de riesgos**: puedes utilizar el análisis de riesgos para decidir cómo y cuándo asignar recursos a las pruebas. Los riesgos son difíciles de determinar con precisión por parte del probador o el desarrollador individualmente; un esfuerzo conjunto de todos los implicados en la aplicación móvil puede ayudar a mejorar el análisis de riesgos. A veces, el probador es el encargado de ajustar la valoración del riesgo.

— **Escalar las pruebas**: para aumentar el volumen y potencialmente la calidad de las pruebas se puede aumentar el rendimiento de estas escalándolas, por ejemplo mediante la automatización de los tests, los sistemas de prueba basados en la nube e involucrando a más personas en el proceso. El uso de herramientas de análisis estático para revisar código y de otros artefactos también puede ayudar al equipo a encontrar y corregir problemas antes de lanzar la aplicación.

Implica a Usuarios Finales en tus Pruebas

Los equipos de desarrollo necesitan un espejo en el que reflejarse para desarrollar una aplicación móvil útil. Un feedback de usuarios tempranero puede proporcionar ese espejo. No necesitas muchos usuarios finales para tener buen feedback[4]. El mayor valor se obtiene con la participación temprana de múltiples usuarios en sesiones regulares y con múltiples pruebas más pequeñas. Los probadores pueden guiar y facilitar las

3 hp.com/go/sprinter

4 nngroup.com/articles/why-you-only-need-to-test-with-5-users

pruebas al usuario final, por ejemplo mediante la preparación de las mismas, el proceso de los archivos de registro y el análisis de los resultados. También pueden volver a probar correcciones hechas en la aplicación.

Cuando terceros están involucrados en probar una aplicación, necesitan maneras de acceder y utilizarla. Las aplicaciones web pueden hospedarse en línea, tal vez protegidas mediante contraseñas, direcciones URL difíciles de adivinar y otras técnicas. Las aplicaciones instalables necesitan por lo menos una forma de ser instaladas, por ejemplo utilizando una tienda de aplicaciones de empresa o servicios de implementación especializados. Para encontrar usuarios finales se puede recurrir a proveedores de servicios crowdsourcing[5].

Cuando la aplicación está más cerca de estar ya lista para producción, los usuarios pueden probar la última versión de la aplicación móvil en fases de tests alfa/beta. Un equipo de desarrollo u organización pueden ofrecer a una comunidad en línea la posibilidad de dar acceso temprano a nuevos lanzamientos, puntos de fidelidad o calificaciones a los usuarios finales. Esta comunidad debe ser un ecosistema amigable para así recibir feedback antes de que la aplicación móvil se envíe a la tienda de aplicaciones.

Prácticas de Testeo Efectivas

Testear, al igual que otras competencias, se puede mejorar mediante la aplicación de diversas técnicas y prácticas. Algunas de ellas tienen que aplicarse durante el desarrollo de tu aplicación móvil, como la capacidad de testeo, mientras que otras se aplican al crear tus pruebas y algunas incluso cuando las estás llevando

5 crowdtesting Applause.com, PassBrains.com y TestBirds.de

a cabo. Testdroid ofrece una buena lista de control[6] de cara a adquirir un buen dominio del testeo por parte de tu equipo.

Mnemónicos que Resumen los Tests Heurísticos

Los heurísticos son directrices o reglas generales que tienden a ser útiles. Se han creado varios específicamente para ayudar a probar aplicaciones móviles y algunos de ellos utilizan códigos mnemónicos para ayudarte a considerar aspectos particulares del software. Cada letra es la inicial de una palabra clave.

— **I SLICED UP FUN**[7]: **I**nput (Prueba la aplicación cambiando su orientación (horizontal/vertical) y probando todas los mecanismos de entrada incluyendo teclado, gestos, etc.), **S**tore (Utiliza las directrices de la tienda de aplicaciones como fuente de ideas para pruebas), **L**ocation (Haz pruebas en movimiento para comprobar si hay problemas de localización), **I**nteraction/Interruption (Mira de qué manera interactúa tu aplicación con otros programas, en particular las aplicaciones nativas del sistema), **C**ommunication (Observa el comportamiento de tu aplicación cuando el dispositivo recibe llamadas, correos electrónicos, etc.), **E**rgonomics (Busca áreas con problemas de interacción, por ejemplo fuentes pequeñas), **D**ata (Prueba el uso de caracteres especiales, diferentes idiomas, fuentes de datos externas, archivos grandes en diferentes formatos, notificaciones), **U**sability (Busca cualquier acción de usuario que sea torpe, confusa o lenta), **P**latform (Prueba sobre diferentes versiones del sistema operativo), **F**unction (Verifica que todas las funciones están implementadas y que funcionan de la manera que se supone que deben hacerlo), **U**ser Scenarios (Crea escenarios de pruebas para tipos

6 testdroid.com/testdroid/6336/get-the-superb-expertise-in-your-testingqa-team

7 kohl.ca/articles/ISLICEDUPFUN.pdf

concretos de usuarios), **N**etwork (Haz pruebas en condiciones de conexión a internet diferentes y cambiantes)

— **COP FLUNG GUN**[8] resume aspectos similares: **C**ommunication, **O**rientation, **P**latform, **F**unction, **L**ocation, **U**ser Scenarios, **N**etwork, **G**estures, **G**uidelines, **U**pdates, **N**otifications.

Implementando la Testeabilidad

Comienza a diseñar e implementar formas de probar tu aplicación ya durante la fase de desarrollo; especialmente en el caso de pruebas automatizadas. Por ejemplo, utilizar técnicas como la inyección de dependencias en el código permite sustituir servidores reales (lentos y fragmentados) con servidores simulados (controlables y rápidos). Utiliza identificadores únicos y claros para los elementos clave de la interfaz de usuario: si los mantienes inalterados, tus pruebas automatizadas requerirán menos mantenimiento.

Separa tu código en módulos que se puedan testear. Hace años, cuando los dispositivos móviles y las herramientas de software eran muy limitadas, los desarrolladores optaron por 'optimizar' su código móvil en bloques monolíticos de código. Sin embargo, los dispositivos y plataformas móviles actuales hacen que esta forma de 'optimización' sea innecesaria y posiblemente contraproducente. Estos dos temas se recogen en un artículo interesante publicado en el Google Testing Blog, Android UI Automated Testing[9].

Proporciona maneras para consultar el estado de la aplicación, posiblemente a través de una interfaz de depuración personalizada. En caso contrario, tus probadores o tú mismo podríais pasar mucho tiempo tratando de comprender qué está fallando cuando la aplicación no funciona como se esperaba.

8 moolya.com/blogs/2014/05/34/COP-FLUNG-GUN-MODEL

9 googletesting.blogspot.co.uk/2015/03/android-ui-automated-testing.html

Tours para Testing Exploratorio

Un tour es un tipo de prueba exploratoria, una manera de estructurar más las sesiones de testing exploratorio. Los tours ayudan a enfocar tus tests, Cem Kaner describe un tour como "...una búsqueda dirigida a través del programa. Encuentra todas las funciones. Encuentra todas las especificaciones sobre el producto. Encuentra todas las variables. Encuentra todos los beneficios previstos. Encuentra todas las maneras de ir de A a B. Encuentra todas las X. O tal vez no TODAS, pero encuentra un buen puñado..."[10]. Con la combinación de diferentes tours en diferentes perspectivas (revisa el heurístico I SLICED UP FUN) se puede escoger la cobertura y profundidad de la prueba.

Algunos ejemplos de tours[11] incluyen los siguientes:

- **Tour de configuración**: intenta encontrar todas las maneras en que se puede cambiar la configuración en el producto de manera que la aplicación conserve dichos ajustes.
- **Tour de funciones**: muévete a través de la aplicación y familiarízate con todos los controles y características que te encuentres.
- **Tour de estructura**: encuentra todo lo que puedas acerca de lo que se refiere al producto físico (código, interfaces, hardware, archivos, etc.).
- **Tour de variabilidad**: busca las cosas que se pueden cambiar en la aplicación y trata de cambiarlas.

10 kaner.com/?p=96; consulta también developsense.com/blog/2009/04/of-testing-tours-and-dashboards/

11 en michaeldkelly.com/blog/2005/9/20/touring-heuristic.html

Personajes

Los personajes (en inglés y derivando del latín, personas) pueden utilizarse para reflejar los distintos usuarios del software. Se pueden diseñar para reflejar o modelar un individuo específico o un conjunto de criterios clave para un grupo de usuarios. Independientemente de la forma en que se creen, cada personaje es singular, no un grupo de personas. Los personajes se pueden utilizar para tener una idea clara de diversos usuarios finales a tener en cuenta de tal manera que se ejecuten pruebas representativas para esos usuarios finales. Existen varios materiales de investigación disponibles en personas.dk.

Testear en Varios Dispositivos

Algunos errores son universales y se pueden descubrir en cualquier dispositivo móvil. Otros muchos aparecen en un subconjunto de los dispositivos, algunos de ellos en dispositivos específicos. Un ejemplo de problemas específicos de dispositivos con Android en Samsung es *verybadalloc.com/android/2015/12/19/special-place-for-samsung-in-android-hell*. Todo esto significa que necesitamos tanto dispositivos en los que probar como probar en varios dispositivos.

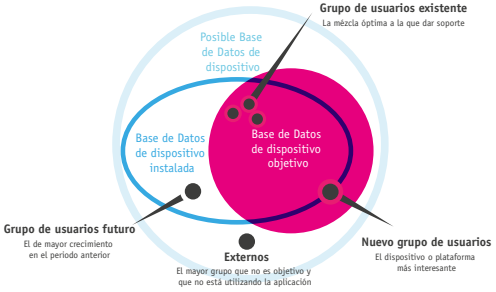
Dispositivos Físicos y Virtuales

Los dispositivos físicos son reales, los puedes sostener en tus manos. Los dispositivos virtuales se ejecutan como software, dentro de otro ordenador. Ambos son útiles para probar las aplicaciones móviles.

Los dispositivos virtuales habitualmente son gratis y están disponibles de manera inmediata para su instalación y uso. Algunas plataformas, incluyendo Android, te permiten crear dispositivos personalizados, por ejemplo con una nueva resolu-

ción de pantalla, que puedes utilizar para probar aplicaciones incluso antes de que un hardware en concreto esté en el mercado. Estos dispositivos virtuales pueden facilitar pruebas básicas rápidas de tus aplicaciones. Las principales diferencias consisten en el rendimiento, la seguridad y la manera en que interactuamos con ellos en comparación con los dispositivos físicos. Estas diferencias pueden afectar a la validez de algunos resultados de las pruebas. Además de los dispositivos virtuales de la plataforma Android, puedes utilizar GenyMotion.com, un emulador de Android más rápido y más capaz, por ejemplo para el control de valores de los sensores.

El conjunto de dispositivos de prueba tiene que ser revisado de manera continua a medida que evoluciona el ecosistema. También es posible que durante tus revisiones identifiques nuevos dispositivos que tu aplicación no soporta actualmente, como se muestra a continuación:



En última instancia, tu software tiene que ejecutarse en dispositivos reales, físicos, tal y como será utilizado por sus usuarios previstos. Las características de rendimiento de distintos modelos de teléfono varían enormemente de unos a otros y también respecto a los dispositivos virtuales. Por tanto, compra, pide prestados y mendiga teléfonos si hace falta para

realizar tus pruebas. Un buen comienzo consiste en elegir una mezcla de dispositivos populares, nuevos y que incluyen características específicas en aspectos como pantalla táctil, teclado físico, resolución de pantalla, conectividad, etcétera. Prueba tu software en al menos uno de gama baja o antiguo, ya que también deseas satisfacer a usuarios con este tipo de dispositivos.

He aquí algunos ejemplos de áreas a testear en dispositivos físicos:

- **Navegación por la interfaz de usuario:** por ejemplo, ¿pueden los usuarios utilizar tu aplicación con una sola mano? Efectos de las diferentes condiciones de iluminación: la experiencia de la interfaz de usuario puede diferir bajo la luz del sol cuando estás fuera de casa. Es un dispositivo móvil, la mayoría de los usuarios estarán en movimiento. Gira la pantalla y asegúrate de que la aplicación es igualmente atractiva y funcional.
- **Localización:** si utilizas la información de ubicación en tu aplicación, muévete tanto de forma rápida como lenta. Ve a lugares con cobertura de red y GPS irregulares para ver cómo se comporta tu aplicación.
- **Multimedia:** el soporte para audio, reproducción de vídeo y servicios de grabación puede diferir drásticamente entre los dispositivos reales y sus respectivos emuladores.
- **Conectividad a internet:** conectar a internet puede llevar una cantidad increíble de tiempo. El retardo en la conexión y el ancho de banda dependen de la red, su potencia real y el número de conexiones simultáneas. Pon a prueba los efectos de una conectividad intermitente para ver cómo responde tu aplicación.

Como ya se ha mencionado, el crowdtesting también puede

ayudar a cubrir un amplio rango de dispositivos reales, pero nunca debes tomar las observaciones de terceros como única referencia.

Dispositivos Remotos

Si no tienes dispositivos físicos a mano o si necesitas probar la aplicación en otras redes, especialmente en el extranjero y para otras localizaciones, puedes utilizar un 'servicio de dispositivos remotos'. Pueden ayudarte a extender la amplitud y profundidad de tus pruebas con poco o ningún coste.

Varios fabricantes ofrecen este servicio sin coste alguno a los desarrolladores de software registrados y para un subconjunto de sus modelos de teléfono. Samsung[12] (para Android y Tizen) proporciona acceso diario restringido pero gratis.

También puedes utilizar los servicios comerciales de empresas como *SauceLabs.com*, *testdroid.com*, *PerfectoMobile.com* o *DeviceAnywhere.com* para realizar pruebas similares a través de una gama de dispositivos y plataformas. Algunos fabricantes promueven estos servicios pero a menudo hay que pagar por ellos después de un corto periodo de prueba. Algunos de estos servicios comerciales proporcionan APIs para que puedas crear pruebas automatizadas.

Puedes incluso crear un repositorio privado de dispositivos remotos, por ejemplo alojándolos en oficinas en localizaciones remotas.

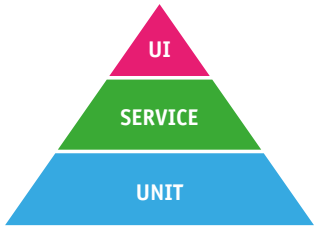
Ten cuidado con las cuestiones de privacidad y confidencialidad cuando utilices dispositivos compartidos.

12 developer.samsung.com/remotetestlab/rtlDeviceList.action

Pruebas Automatizadas

Las pruebas automatizadas pueden ayudarte a mantener y mejorar tu velocidad, es decir tu capacidad de lanzar funcionalidades, gracias a que proporcionan detección temprana de problemas. Para ello tienen que estar bien diseñadas e implementadas. Las pruebas automatizadas buenas imitan las buenas prácticas de desarrollo de software, por ejemplo utilizando patrones de diseño[13], modularización, llevando a cabo revisiones de código, etcétera. Para poder automatizar es necesario tener una cierta habilidad escribiendo código, mayor o menor en función de la herramienta escogida. Conviene tomar en consideración las herramientas de automatización de pruebas proporcionadas como parte del SDK de desarrollo. Normalmente son gratis, están inherentemente disponibles para la plataforma en particular y son apoyadas por empresas grandes. La automatización de pruebas puede realizarse a diferentes niveles (echa un vistazo a la imagen con la pirámide de automatización a continuación). Supone una elección estratégica decidir qué debería ser automatizado en las pruebas unitarias, qué a nivel de servicio o de API y qué escenarios a nivel de interfaz de usuario de la aplicación. La pirámide representa la confianza que se va construyendo desde las pruebas unitarias

13 en.wikipedia.org/wiki/Design_Patterns



hasta los niveles superiores. Se necesitan varios niveles de pruebas para demostrar que la aplicación funciona.

Automatización de Pruebas a Nivel de GUI

El primer nivel de automatización lo conforman las pruebas que interactúan con la aplicación a través de la interfaz gráfica de usuario (GUI). Es uno de los elixires de la industria de pruebas: muchos lo han intentado pero pocos han tenido éxito. Una de las principales razones por las que la automatización de pruebas GUI es tan complicada es que la interfaz de usuario está sujeta a cambios significativos que puedan romper la forma en que las pruebas automatizadas interactúan con la aplicación.

Para que las pruebas sean eficaces a largo plazo, y en tanto que la aplicación va cambiando, los desarrolladores necesitan diseñar, implementar y soportar las etiquetas y demás elementos utilizados por las pruebas automatizadas de interfaz. Tanto Apple, con UI Automation[14], como más recientemente Android[15] utilizan la etiqueta Accessibility asignada a elementos de la GUI a modo de interfaz de facto para la automatización de pruebas.

Algunas empresas han convertido sus herramientas a código abierto, por ejemplo appium de SauceLabs[16] y Calabash de Xamarin[17]. Estas herramientas tienen como objetivo proporcionar soporte multiplataforma, particularmente para Android e iOS. Otro framework exitoso de código abierto es Robotium[18] que

14 developer.apple.com/library/tvos/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/UIAutomation.html

15 developer.android.com/tools/testing/testing_ui.html

16 https://github.com/appium/appium

17 github.com/calabash

18 github.com/robotiumtech/robotium

ahora ofrece un producto comercial (un grabador de pruebas). Varias otras herramientas han terminado desapareciendo, ¿tal vez la industria está madurando de tal modo que solamente sobreviven las ofertas más fuertes?

Automatización de Pruebas a Nivel de Servicio

Hay una gran cantidad de lógica de negocio implementada dentro de una API. Los cambios en esta lógica o en el sistema backend pueden ser monitorizados con pruebas automatizadas de API. El objetivo de las pruebas puede ser la regresión funcional o también la fiabilidad, el rendimiento y la seguridad. Para las pruebas de regresión funcional es útil una herramienta como Postman[19].

Varias herramientas pueden ser de ayuda con las pruebas de APIs. Por ejemplo Fiddler de Telerik[20] y Charles Proxy[21]. Ambas te permiten ver y modificar el tráfico de red entre tu dispositivo móvil y la red.

Automatización de Pruebas a Nivel Unitario

Las pruebas unitarias consisten en escribir pruebas automatizadas que ponen a prueba pequeñas secciones de código, habitualmente solo unas pocas líneas. Generalmente deben ser escritas por el mismo desarrollador que escribe el código fuente de la aplicación, ya que reflejan cómo se espera que se comporten esas secciones. Las pruebas unitarias gozan de un largo pedigrí en el desarrollo de software, donde JUnit[22] ha dado lugar a frameworks parecidos para prácticamente

19 blog.getpostman.com/2014/03/07/writing-automated-tests-for-apis-using-postman/

20 telerik.com/fiddler

21 charlesproxy.com/documentation/getting-started

22 en.wikipedia.org/wiki/JUnit

todos los lenguajes de programación utilizados para desarrollar aplicaciones para móviles.

Automatización de Pruebas en BDD

BDD es un enfoque de desarrollo guiado por comportamiento (Behavior-Driven Development en inglés) dentro de la familia del TDD o desarrollo guiado por tests (Test-Driven Development en inglés)[23]. El comportamiento se describe en archivos de texto con un formato específico que permite que estos archivos sean ejecutados como pruebas automatizadas. El formato de las pruebas tiene la intención de ser legible y comprensible por cualquier persona involucrada en el proyecto de software. Pueden ser escritas en casi cualquier idioma humano, por ejemplo japonés[24], y utilizan una estructura consistente, sencilla, con frases como **Dado que, Cuando, Entonces** para estructurar las pruebas.

El principal framework de BDD para testear aplicaciones móviles es Calabash para Android e iOS[25]. Algunos otros han dejado de desarrollarse o mantenerse durante el último año y pueden considerarse muertos para todos (excepto para los desarrolladores más persistentes). Los frameworks BDD de propósito general pueden ser relevantes cuando se integran con otros frameworks como appium, que utiliza el protocolo WebDriver (un estándar W3)[26].

La automatización también puede ayudar a las pruebas manuales, por ejemplo, sustituyendo los pasos manuales más propensos a errores cuando se está testeando, o reduciendo el tiempo y el esfuerzo necesarios para automatizar una serie de

23 en.wikipedia.org/wiki/Behavior-driven_development

24 github.com/cucumber/cucumber/tree/master/examples/i18n/ja

25 github.com/calabash

26 w3.org/TR/webdriver

capturas de pantalla. Los desarrolladores pueden ayudar a los probadores a ser más eficientes proporcionándoles herramientas automatizadas, por ejemplo, despliegue de aplicación a través de ADB[27].

Testear las Cinco Fases del Ciclo de Vida de una App

El software se desarrolla en fases llamadas fases del ciclo de vida. Un probador de la aplicación móvil puede ser parte del equipo de desarrollo, pero también puede ser responsable de facilitar las pruebas de la experiencia del usuario en producción. Hay diferentes tareas a realizar dependiendo de en qué fase o fases estés involucrado en el ciclo de vida. Por ejemplo, al unirte a un equipo de desarrollo tu tarea puede ser el análisis de errores en los archivos de registro en un dispositivo. En cambio, al unirte a una fase de pruebas beta una tarea puede ser el análisis de los resultados de las pruebas de usabilidad, como la grabación de vídeo. El ciclo de vida de una aplicación móvil encaja en 5 fases: implementación, verificación, lanzamiento, retención y validación.

Ciclos de Mejora

Las pruebas se aplican en cada fase. Algunas de las decisiones que se toman en las etapas previas pueden afectar tus pruebas en etapas posteriores. Por ejemplo, si decides que quieres pruebas automatizadas del sistema en la primera fase, serán más fáciles de aplicar después en las fases posteriores. El concepto de cinco fases podría sugerir que van una tras otra y forman un flujo lógico como agua río abajo. No es el caso. Cada paso en las diferentes fases ofrece la posibilidad de aprender y mejorar.

27 thefriendlytester.co.uk/2015/11/deploying-to-multiple-android-devices.html

Al llevar a cabo los tests, el equipo aprende tanto lo buena que es la aplicación móvil como las áreas de mejora en la forma en que se produce la aplicación. El desarrollo de aplicaciones móviles es una actividad desafiante, compleja y dinámica que no funciona perfectamente a la primera, por lo que los equipos deben incorporar un ciclo de mejora para que puedan aprender y mejorar activamente lo que hacen.

Fase 1: Implementación

Esta fase incluye diseño, código, pruebas unitarias y creación de tareas. Tradicionalmente, los probadores no están involucrados en estas tareas. Sin embargo, un buen testeo aquí puede mejorar significativamente la calidad y el éxito de la aplicación al ayudarnos a estar seguros de que nuestra implementación se hace correctamente.

En términos de pruebas, deberías decidir sobre las siguientes cuestiones:

- ¿Utilizar desarrollo guiado por pruebas (TDD)?
- Ayuda a revisar los diseños sobre los flujos de usuario principales, alternativos y negativos.
- ¿Qué datos de pruebas utilizas para validar los flujos de usuario?
- ¿Tendrás sistemas automatizados de pruebas? En caso afirmativo, ¿cómo facilitarás dichas pruebas? Por ejemplo, añadiendo etiquetas adecuadas a objetos clave en la interfaz de usuario.
- ¿Cómo validarás tus aplicaciones? ¿A través del uso de analíticas móviles, por ejemplo? ¿Con informes de errores? ¿Con el feedback de los usuarios?

Cuestiona el diseño. Tienes que asegurarte de que cumple con los fines previstos y de que no se están cometiendo er-

rores graves. El documento de Phillip Armour 'Five Orders Of Ignorance'[28] es un gran recurso para ayudar a estructurar tu enfoque.

Fase 2: Verificación

Revisa las pruebas unitarias, las pruebas de instalación interna y las pruebas del sistema y evalúa su potencia: ¿son realmente útiles y dignas de confianza? Nota: también deben ser revisadas como parte de la fase de implementación; sin embargo este es un buen momento para abordar las deficiencias materiales antes de que el desarrollo se considere 'completo' para la base de código actual.

Para las aplicaciones que requieren instalación, necesitas maneras de desplegarlas en dispositivos específicos para las pruebas previas al lanzamiento. Para algunas plataformas (incluyendo Android, iOS y Windows), los teléfonos deben estar configurados para que las aplicaciones en fase de desarrollo se pueden instalar. En función de tu estrategia de pruebas puedes decidir en qué teléfonos, plataformas, versiones y resoluciones quieres basar el alcance de las pruebas y del soporte.

Las pruebas del sistema se realizan a menudo de forma interactiva, por parte de los probadores. También tienes que considerar cómo asegurarte de que la aplicación cumple con:

- La usabilidad, experiencia de usuario y requisitos estéticos.
- El rendimiento, especialmente del modo en que lo percibirán los usuarios finales[29].
- Las pruebas de internacionalización y localización.

28 www-plan.cs.colorado.edu/diwan/3308-07/p17-armour.pdf

29 Una herramienta de pruebas de rendimiento relevante es ARO (Application Resource Optimizer) de AT&T: | developer.att.com/application-resource-optimizer, proyecto de código abierto en github.com/attdevsupport/ARO

Fase 3: Lanzamiento

Aquellos de vosotros que aún no habéis trabajado con las principales tiendas de aplicaciones, estad preparados para una desafiante experiencia donde la mayoría de los aspectos están fuera de tu control, incluyendo los plazos para la aprobación de tu aplicación. Además, en algunas tiendas de aplicaciones no es posible revertir a una versión anterior, así que si tu versión actual tiene defectos importantes, tienes que crear una nueva versión que corrija los defectos y, a continuación, esperar hasta que haya sido aprobada por la tienda de aplicaciones antes de que tus usuarios puedan recibir una versión operativa de la aplicación.

Dadas estas limitaciones es útil ampliar tus pruebas para incluir verificaciones previas a la publicación y pruebas beta de la aplicación como por ejemplo si es adecuada para el conjunto de los dispositivos de destino y para los usuarios finales. Los proveedores de las principales plataformas publican actualmente pautas para ayudarte a verificar que tu aplicación responde a sus criterios de publicación. Estas directrices pueden ayudarte incluso si te diriges a otras tiendas de aplicaciones, y pueden ser utilizadas a modo de lista de comprobaciones durante la fase de implementación.

Apple	developer.apple.com/appstore/resources/approval/guidelines.html
Android	developer.android.com/distribute/googleplay/publish/preparing.html#core-app-quality
Windows Phone	msdn.microsoft.com/en-us/library/windowsphone/develop/hh394032
BlackBerry	developer.blackberry.com/devzone/appworld/tips_for_app_approval.html

Fase 4: Vinculación

Esto incluye búsqueda, confianza, descarga e instalación. Una vez que tu aplicación está disponible públicamente, los usuarios necesitan encontrarla, confiar en ella, descargarla e instalarla. Puedes probar cada aspecto de esta fase antes y cuando ya está en producción. Intenta buscar tu aplicación en la tienda de aplicaciones pertinente y en los motores de búsqueda principales. ¿De cuántas maneras diferentes puede ser encontrada por los usuarios a los que va dirigida? ¿Qué pasa con los usuarios fuera de los grupos objetivo, quieres que la encuentren? ¿Cómo van a confiar los usuarios en tu aplicación lo suficiente como para descargarla y probarla? ¿Realmente tu aplicación necesita tantos permisos? ¿De qué tamaño es la descarga y puede eso suponer un problema para descargarla mediante la red móvil? ¿Cabrá en el teléfono del usuario, especialmente si hay poco espacio de almacenamiento disponible en su dispositivo? ¿Se instala correctamente la aplicación? Puede que haya problemas de firmado que causen que sea rechazada por algunos dispositivos.

Fase 5: Validación

Esto incluye el pago, el uso y el feedback. Como seguramente ya sabes, es poco probable que una aplicación móvil con poco feedback tenga éxito. Además, muchas de las aplicaciones tienen una vida muy corta en el teléfono de un usuario. Si la aplicación no es de su agrado ni le involucra en pocos minutos, es probable que sea descartada o ignorada. Y si estás buscando obtener pagos, es buena idea probar las diferentes formas de pago (sobre todo los integrados en la aplicación).

Intenta encontrar maneras de probar los siguientes puntos tan pronto como te sea posible:

— Detección y reporte de problemas. Pueden incluir tu propio código, utilidades de terceros y servicios en línea.
— Analíticas móviles. ¿Tienen sentido los datos recabados? ¿Qué anomalías hay en los datos reportados? ¿Cuál es la latencia en la recepción de resultados?

Aprender Más

Testear aplicaciones móviles se está convirtiendo en una corriente principal con múltiples y buenas fuentes de información. Algunos recursos útiles son los siguientes:

— *blog.testmunk.com* el blog de Testmunk tiene un amplio rango de artículos relevantes sobre el testeo de aplicaciones móviles.
— *enjoytesting.files.wordpress.com/2013/10/mobile_testing_ready_reckoner.pdf* contiene ideas cortas y claras sobre pruebas, con ejemplos incluidos básicamente para dispositivos Android.
— *developers.google.com/google-test-automation-conference/2015/presentations* En el año 2015, la conferencia Google Test Automation Conference (GTAC) incluyó al menos 5 presentaciones relacionadas con el testeo de aplicaciones móviles, merece la pena mirarlas.
— *handsonmobileapptesting.com/* que enlaza con el libro: Hands-on Mobile App Testing, de Daniel Knott. Se trata de un libro bien escrito que trata sobre varios aspectos del testeo de aplicaciones móviles. En el sitio web está disponible un capítulo de muestra.
— *testdroid.com/blog*, un blog muy poblado sobre varios temas, incluyendo el testeo de aplicaciones móviles.

También tienen una serie sobre testeo de juegos para móviles[30].

- *appqualitybook.com*, el sitio web sobre el interesante libro de Jason Arbon basado en las experiencias de testear y analizar un gran número de aplicaciones móviles.
- *appqualityalliance.org/resources*, el sitio web oficial de App Quality Alliance AQuA que incluye sus útiles directrices sobre testeo de aplicaciones.

30 testdroid.com/testdroid/7790/best-practices-in-mobile-game-testing





BY Julian Harty

Analíticas Móviles

¿Te gustaría saber más acerca de cómo se está utilizando tu aplicación y lo bien que se está comportando? Si es así, las analíticas móviles pueden ser tus aliadas y ayudarte a entender las maneras en que tu aplicación se está utilizando ahí afuera por usuarios reales. Las analíticas móviles encajan con otras fuentes de información, incluyendo calificaciones de las tiendas de aplicaciones, analíticas de errores fatales, pruebas de multitud y estudios de usabilidad.

Los datos de las analíticas móviles pueden ser de ayuda en muchos aspectos de nuestro trabajo, incluyendo aspectos de negocio, sociales, de operaciones y tecnológicos. Los datos capturados se pueden utilizar para orientar tu trabajo y reducir las ineficiencias. Estarías en buena compañía, en tanto que la gran mayoría de las 500 aplicaciones de más éxito tanto en iOS como en Android incluyen al menos una librería de analíticas móviles, según datos de SourceDNA[1].

La galaxia móvil presenta una riqueza increíble en la que tu software se puede utilizar en muchos dispositivos extraños que presentan diferencias significativas en su comportamiento. Algunos investigadores descubrieron que la descarga de la batería podía producirse tres veces más rápidamente cuando se utilizaba su aplicación en dispositivos con especificaciones de hardware similares y, entre otras cosas, descubrieron una aplicación que estaba utilizando código específico para reducir el brillo de la pantalla al ser ejecutada en un dispositivo Kindle Fire para mejorar la vida de la batería en un 40% y de este modo alargar la duración de la sesión de manera significativa.

1 sourcedna.com/stats/

También podemos aprender maneras de mejorar las formas en las que desarrollar y probar el software. Pero pese a la emoción, debemos recordar que hay que proteger la privacidad del usuario y respetar sus preferencias y expectativas. Los efectos de las analíticas móviles pueden molestar a los usuarios por estar consumiendo recursos valiosos, o por abusar de cierta información confidencial sobre el usuario y el uso que hace de la aplicación.

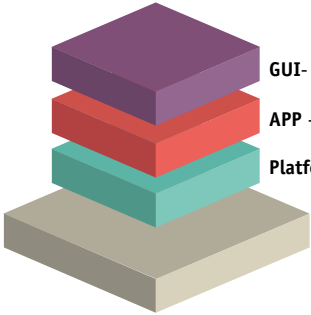
Analíticas para Cada Capa de una Aplicación Móvil

Conceptualmente una aplicación consta de varias capas que se construyen una encima de la otra. La capa superior es la interfaz de usuario que se comunica con el usuario. Prácticamente todas las aplicaciones incluyen una interfaz de usuario gráfica (GUI) que se visualiza en la pantalla del dispositivo. Puede haber otras interfaces de usuario, por ejemplo para capturar el movimiento o un audio o vídeo. La siguiente capa contiene la lógica, lo que la aplicación **hace**. A menudo hay también alguna especie de capa de comunicación. Y en el nivel más bajo está el dispositivo físico con el sistema operativo o plataforma instalados, que soporta y proporciona el entorno de ejecución para la aplicación.

Existen varios tipos de analíticas disponibles que, hasta cierto punto, se pueden sobreponer unas a otras.

La forma más popular de analíticas de la interfaz gráfica de usuario se basa en mapas de calor, que son especialmente adecuados para capturar datos sobre cómo se está utilizando esta interfaz gráfica de usuario. Los mapas de calor se habilitan mediante la incorporación de software en una aplicación para realizar un seguimiento de todas las interacciones del usuario

con la interfaz gráfica de la misma. Hay decenas de opciones comerciales disponibles, Appsee proporciona un servicio particular pulido y ofrece muchos recursos libres incluyendo



libros electrónicos[2] sobre mapas de calor y temas relacionados.

De forma aislada, los datos de un mapa de calor se pueden utilizar para realizar un seguimiento de los "viajes" de un usuario individual a través de la interfaz gráfica de usuario. De forma conjunta, varias analíticas relacionadas con la experiencia del usuario (UX) se pueden deducir a partir de los datos, incluyendo las áreas problemáticas de la interfaz gráfica de usuario.

No parece que haya muchos servicios equivalentes para capturar otras formas de entrada de información, por lo que puede que tengas que escribir tu propio código si quieres o necesitas recopilar datos sobre otras interfaces de usuario.

Las analíticas móviles dentro de la aplicación se adaptan a la capa de la lógica y pueden grabar también algunos detalles de la capa superior (la GUI) y la inferior (el dispositivo).

2 appsee.com/ebooks

Herramientas para Analíticas

Existen como mínimo 20 empresas que ofrecen un festín de soluciones para analíticas móviles con múltiples sabores, desde la monitorización de campañas hasta la mejora de la calidad del software. Muchas además incluyen funcionalidades extra, como informes de errores fatales o la monitorización de usuarios y beneficios. Casi la mitad ofrecen implementaciones de código abierto para sus librerías, ¿tal vez para alejar los temores respecto a su comportamiento?[3]

Muchos de los proveedores de soluciones de analíticas móviles incluyen un apartado 'Getting Started' donde aprender cómo comenzar a utilizar sus productos. Algunos ejemplos son Flurry[4] y KISSmetrics[5]. Suele ser necesario registrarse antes de utilizar estos productos, ya que la mayoría necesitan ser configurados con una 'clave' única para tu aplicación.

Examina muchas de las soluciones potenciales antes de decidirte por una. Lee la documentación y el código de ejemplo para ver cómo de fácil es implementarla en tu aplicación, y revisa las licencias que usa, incluyendo los términos de privacidad. A partir de ahí, elige al menos una para experimentar con ella implementando analíticas móviles en tu aplicación. Integrando su código aprenderás mucho más acerca de lo que puedes conseguir y también acerca de cómo funcionan las analíticas móviles en la práctica. Investiga qué soluciones utilizan las demás aplicaciones y por qué lo hacen. Por ejemplo, VentureBeat descubrió que el 95% de los desarrolladores Android utilizan Google Analytics, pero "a pesar de

3 readwrite.com/2013/12/05/why-mobile-developers-need-open-source-analytics-embedded-in-their-applications

4 support.flurry.com

5 support.kissmetrics.com/getting-started/overview

la enorme cuota de mercado de Google, menos de una tercera parte de los desarrolladores móviles la considera su solución principal para analíticas móviles"[6]. Y otro caso más, Twitter Answers ha crecido desde la nada hasta convertirse en el líder del mercado en 2015[7]), un ejemplo de lo rápido que el mercado se está transformando.

Si trabajas con aplicaciones multiplataforma, probablemente desees consistencia a través de cada una de ellas. De otra manera puedes encontrarte intentando comparar paquetes de datos que difieran o sean incluso completamente diferentes, especialmente si utilizas diferentes soluciones de analíticas móviles para cada plataforma. Así pues, considera elegir una única solución común que soporte todas las plataformas en las que deseas publicar tu aplicación.

Hay dos proveedores que merece la pena estudiar. Segment.io[8] abstrae un amplio abanico de métricas. Su código abierto[9] disminuye el esfuerzo necesario para adaptarse a diferentes proveedores de analíticas. *Count.ly*[10] proporciona implementaciones de código abierto tanto de su librerías de servidor como de cliente y animan a los usuarios a crear un entorno de pruebas completo para evaluar su producto.

Decidir Qué Medir

¿Qué te gustaría medir para comprender cómo se está utilizando la aplicación? He aquí algunas sugerencias:

6 venturebeat.com/2014/12/02/230-developers-and-1-8m-apps-reveal-the-best-mobile-app-analytics-solutions/

7 sourcedna.com/stats/

8 segment.io

9 github.com/segmentio

10 count.ly

- **Eventos de uso clave:** por ejemplo, la nueva opción de búsquedas o cuándo abren los usuarios una red social desde tu aplicación.
- **Eventos centrados en el negocio:** cualquier interacción generada por el usuario que suponga un beneficio para ti. ¿Con qué frecuencia compran tus usuarios la versión premium de tu aplicación y otros ítems ofrecidos en su interior? ¿En qué momento cancelan pedidos o descartan un carrito de compra antes de realizar el pago?
- **Eventos centrados en la aplicación:** rendimiento, usabilidad, fiabilidad y otros datos sobre el comportamiento de la aplicación.

Una vez definidas las principales áreas de interés tendrás que diseñar las métricas de análisis, por ejemplo qué partes de los datos es necesario reportar.

Definir Cómo Medir

Crea nombres significativos para tus eventos de interacción, de tal modo que puedas recordar fácil y correctamente lo que están midiendo. Decide qué elementos necesitan ser incluidos para cada evento que desees grabar. Considera cómo serán utilizados los datos una vez que se hayan recogido; por ejemplo, esboza informes y diagramas estándar y establece

cómo se van a procesar los diversos datos para generar cada informe o gráfico.

Del mismo modo, recuerda abordar temas sobre globalización tales como la fecha y hora de cada elemento. ¿La aplicación detecta la fecha y hora de un evento de acuerdo a la ubicación del dispositivo, a su configuración o utilizando un tiempo global como el UTC?

Muchas de las soluciones de analíticas móviles grabarán y reportarán automáticamente los elementos de datos al servidor. Vale la pena comprobar cuáles son estos elementos, cómo y cuándo se presentan, y cómo están formateados. A partir de ahí podrás decidir si deseas utilizar y confiar en estos elementos reportados automáticamente.

Las etiquetas personalizadas de eventos aumentan los eventos predefinidos. Muchas de las soluciones de analíticas móviles proporcionan medios para que tu aplicación los genere. Es posible que necesites formatear los mensajes de eventos personalizados. Si es así, presta atención a la codificación de los elementos y los separadores. Es posible, por ejemplo, que deban utilizar codificación URL[11] cuando se envían como mensajes REST.

Puede que desees considerar la frecuencia con que la aplicación debe informar sobre eventos para así minimizar el riesgo de saturar la capacidad disponible del sistema de analíticas, lo que podría afectar a la fiabilidad y exactitud de los datos proporcionados. Localytics tiene algunos ejemplos de integración interesantes[12]. Un método para reducir el volumen de datos procesados por las soluciones de analíticas es el

11 en.wikipedia.org/wiki/Percent-encoding

12 support.localytics.com/Integration_Overview

muestreo. Adam Cassar ha publicado un post interesante sobre este tema[13].

Configurar tu Aplicación

Puede que tengas que declarar capacidades adicionales necesarias para que las analíticas móviles funcionen correctamente cuando se integren con tu aplicación.

Para Android, se las conoce como permisos. Las analíticas probablemente necesitarán permiso de acceso a internet de modo que los eventos puedan ser enviados en línea, y también permisos para conocer la localización en el caso de que la solución registre la ubicación del teléfono. Si tu aplicación ya está utilizando estos permisos no es necesario especificar su uso de nuevo.

Para iOS, `UIRequiredDeviceCapabilities` le indica a iTunes y a la App Store qué características de dispositivo necesita la aplicación. Se implementa como un diccionario donde los elementos se especifican utilizando claves, que incluyen WiFi, servicios de localización y GPS.

Windows Phone 8.1 y 10 utilizan App Specific Hardware ID (ASHWID)[14].

Gestionar los Resultados

Hay un lapso de tiempo entre que una aplicación envía un evento de analíticas y la información es procesada y puesta a tu disposición. El retardo, o latencia, varía entre muy cerca de

13 periscopix.co.uk/blog/should-you-be-worried-about-sampling

14 msdn.microsoft.com/en-us/library/windows/apps/jj553431.aspx y msdn.microsoft.com/EN-US/library/windows/apps/windows.system.profile.hardwareidentification.getpackagespecifictoken.aspx

'tiempo real' y varias horas. Tú, y los promotores de tu negocio, debéis decidir cuánto tiempo podéis permitiros retrasar el conocimiento sobre acontecimientos en tiempo real.

Algunas soluciones de analíticas proporcionan una API para acceder a los datos. Esto te permite tener un mayor campo de visión a la hora de crear informes personalizados. Algunos te permiten hospedar los servidores, lo que te aporta mayor control sobre los datos y su uso.

Para evaluar la calidad de los resultados, algunas empresas invierten el esfuerzo adicional en incorporar múltiples soluciones de análisis en su aplicación y cruzar después los resultados. Sin embargo, dos resultados contradictorios no se fáciles de reconciliar, así que puede ser necesario el uso de tres bloques de resultados para diagnosticar las diferencias por triangulación[15].

KISSmetrics proporciona consejos prácticos sobre cómo probar si la implementación está funcionando[16].

15 en.wikipedia.org/wiki/Triangulation_(social_science)

16 support.kissmetrics.com/getting-started/testing-km

¿Qué Puede Ir Mal?

El camino al infierno está empedrado de buenas intenciones. Hay muchas cosas que pueden salir mal en la implementación de analíticas móviles. Estas son algunas de los más comunes:

- **Resultados sin calibrar**: confiar ciegamente en los datos puede conducir a una vorágine de problemas. Los resultados puedes ser inexactos y engañosos, causando una ronda de problemas si se utilizan para gestionar tu negocio y trabajo. Una buena práctica es poner a prueba la aplicación de las analíticas desde el principio, empezando sin usuarios, y probando después con solo uno antes de probar con más. Revisa la latencia, la precisión y la fiabilidad de los datos capturados.
- **Traicionar la confianza**: los usuarios por defecto confían que las aplicaciones se comportarán correctamente en sus dispositivos móviles. Sin embargo, las aplicaciones o los SDKs pueden romper accidental o deliberadamente esa confianza, por ejemplo mediante el seguimiento de los usuarios, la grabación y uso de datos sensibles, etc. Trata de no esconderte detrás de licencias de uso que poca gente lee y menos gente aún entiende. En su lugar, asegúrate de que tu aplicación y las librerías de analíticas que utilices se comportan de forma adecuada y "haz lo que quisieras que te hicieran a ti y no husmees donde no debes".
- **Regalar el tesoro**: asegúrate de que tienes derechos y acceso suficientes a los datos que se obtienen por parte del software de analíticas. Esto es especialmente relevante cuando se utilizan las librerías y servicios de terceros.

Ten presente que algunos proveedores de soluciones de analíticas móviles pueden utilizar los datos reportados por tu aplicación y proporcionárselos y vendérselos a otros. Pueden controlar la existencia de los datos, lo que significa que podrían hacer que sean inaccesible para ti. Del mismo modo, pueden conservarlos y utilizarlos mucho después de que hayas retirado tu aplicación. Si hay información de tipo personal en los datos puede haber implicaciones legales y de privacidad adicionales.

SafeDK es una startup reciente que se centra en el comportamiento de los SDKs, incluyendo los de analíticas móviles. Los SDKs añadidos a las aplicaciones pueden afectar negativamente a su rendimiento, seguridad y fiabilidad, además de otros problemas y preocupaciones. En el blog de SafeDK[17] se discuten estas consideraciones y se proporcionan consejos sobre cómo seleccionar SDKs entendiendo el comportamiento que muestra cada uno de ellos.

Recuerda que debes explicar a los usuarios finales que la aplicación está diseñada para registrar y compartir información acerca de su uso, a ser posible en los términos y condiciones. Puede que necesites o desees que los usuarios tengan la capacidad de decidir si permiten un seguimiento de su uso de la aplicación. Si es así, facilita al usuario el control sobre los ajustes y considera la posibilidad de dar al usuario una forma de acceder a los datos registrados, borrarlos, o ponerse en contacto con el proveedor de soluciones de analíticas.

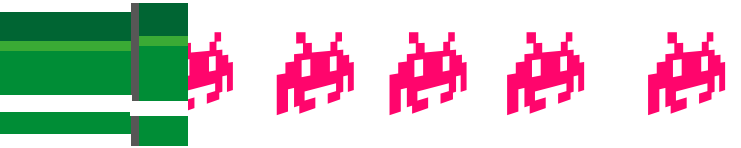
Los proveedores de librerías de terceros parecen tener variadas aproximaciones en lo que se refiere a la privacidad. Algunos declaran que la privacidad de los usuarios es de suma importancia y resaltan la relevancia de no realizarles seguimiento. Google Analytics prohíbe de manera feha-

17 blog.safedk.com

ciente monitorizar información personal que permita la identificación en sus términos de servicio[18]. Otros facilitan ejemplos, incluyendo fragmentos de código, que demuestran cómo grabar datos de identificación personal. Por ejemplo, KISSmetrics provee el siguiente trozo de código[19]: `] identify:@"name@email.com"]`. Mixpanel ofrece un ejemplo de cómo monitorizar usuarios específicos[20].

Hay varios lugares para aprender más sobre temas de privacidad y la ética de trabajar con datos relativos a los usuarios, por ejemplo:

- El post en el blog de Jeff Northrop sobre analíticas móviles[21]
- El libro de Kord Davis "Ethics of Big Data" (O'Reilly, 2012)[22]



18 google.com/analytics/terms/us.html

19 support.kissmetrics.com/apis/objective-c

20 mixpanel.com/activity-feed/

21 jnorthrop.me/privacy-considerations-with-mixpanel-people-analytics

22 disponible en shop.oreilly.com/product/0636920021872.do

Aprender Más

Esperamos que este capítulo haya despertado tu curiosidad por saber más acerca de las analíticas móviles. A continuación, algunos lugares para comenzar tu investigación:

— Varios artículos de Michael Wu o Lithium Technologies. Un buen punto para comenzar es el artículo "Are Your Big Data Analytics Actionable?"[23].
— Capturing Mobile Experience in the Wild: A Tale of Two Apps[24], un estudio de la Universidad de Wisconsin resaltando la importancia de los datos recogidos sobre más de un millón de usuarios a lo largo de tres años utilizando analíticas centradas en la aplicación.
— The Beginner's Guide To App Analytics[25], disponible como descarga gratuita.
— The Mobile Developer's Guide to the Parallel Universe[26], un libro hermano a este mismo que cubre las analíticas móviles desde la perspectiva del marketing.

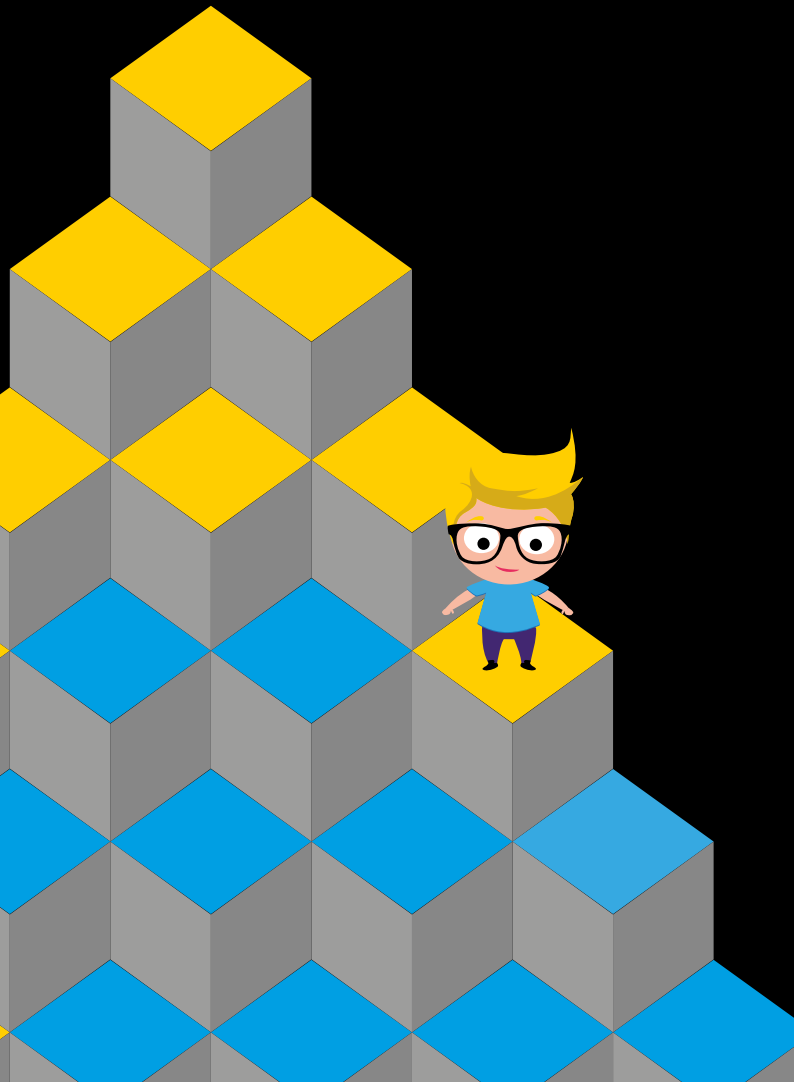


23 community.lithium.com/t5/Science-of-Social-blog/Are-Your-Big-Data-Analytics-Actionable/ba-p/129029

24 static.googleusercontent.com/media/research.google.com/en//pubs/archive/41590.pdf

25 info.localytics.com/download-beginners-guide-to-app-analytics

26 wip.org/resources/#mobile-developers-guide-parallel-universe



BY Julian Harty

Recopilación y Comprensión del Feedback de Usuario

En las aplicaciones móviles, muchos usuarios proporcionan feedback voluntariamente como se puede confirmar en las tiendas de aplicaciones, con algunas apps recibiendo miles de comentarios diarios. Las tiendas de aplicaciones han cambiado sutilmente el equilibrio de poder hacia los usuarios, que ahora tienen unos foros públicos y bien conocidos donde pueden expresar sus quejas y preocupaciones, así como cualquier elogio. Su feedback puede ayudarnos a identificar problemas potenciales de forma rápida, de tal manera que puedan ser "cortados de raíz" o, lo que es lo mismo, arreglados rápidamente antes de que supongan un grave problema. Por otra parte, el feedback puede incluir un montón de sugerencias y recomendaciones potencialmente relevantes que se pueden utilizar para mejorar la aplicación móvil.

El feedback puede complementar las analíticas móviles. Las analíticas móviles proporcionan feedback orientado al software, consulta el capítulo dedicado en esta guía para aprender más sobre tus opciones en esa área. Este capítulo se centra en el feedback creado por humanos.

Valoraciones en Tiendas de Aplicaciones y Feedback

Las tiendas de aplicaciones proporcionan un foro público para que los usuarios valoren las aplicaciones y dejen comentarios por escrito. Como muchos de nosotros sabemos, las aplica-

ciones con puntuaciones bajas son mucho menos propensas a ser descargadas por los usuarios. Por otra parte, las tiendas de aplicaciones parecen utilizar las valoraciones como un factor para decidir la prioridad a la hora de incluir una aplicación en los resultados de las búsquedas, e incluso para promocionar algunas de las aplicaciones más populares. Una excelente lectura sobre este tema es el libro "App Quality Book" de Jason Arbon[1].

Maneras de Recoger Feedback

Varios servicios comerciales ofrecen maneras de reducir el esfuerzo de recopilación y análisis de la información de las tiendas de aplicaciones. Algunos ejemplos son *appannie.com* y *appfigures.com*. Explicaciones útiles para preparar tus propios análisis se pueden encontrar en un par de mensajes de blogs relacionados[2]. Hay varias complicaciones y limitaciones a la hora de recoger feedback de tiendas de aplicaciones específicas, como por ejemplo límites en el número de resultados disponibles. En muchos casos puedes idear maneras para recoger las opiniones. Alternativamente una opción híbrida consiste en utilizar servicios como AppFigures[3].

Tratar con Inconsistencias

En algunos casos, la puntuación que da el usuario y sus observaciones pueden contradecirse entre sí. La gente puede no entender el sistema de puntuación, donde las tiendas de aplicaciones consideran 5 estrellas la calificación más alta. Los usuarios pueden asumir que 1 estrella es la calificación más

1 appqualitybook.com

2 blog.scottlogic.com/2014/03/20/app-store-analysis.html y shinobicontrols.com/news/a-statistical-comparison-of-the-ios-and-android-stores

3 docs.appfigures.com/api/reference/v2/reviews

alta. En otros casos, las puntuaciones pueden parecen estar hechas casi al azar.

En un principio podríamos considerar la prioridad de las puntuaciones a base de estrellas con una escala lineal, en la que 1 estrella es la más baja y la "peor" calificación. Sin embargo, las investigaciones de Spotify y otros descubrieron que los usuarios daban puntuaciones de 2 estrellas para los problemas más serios e importantes. Publicaron sus hallazgos en un artículo llamado "What Do Mobile App Users Complain About?"[4].

Feedback Deshonesto

El feedback deshonesto es el feedback enviado deliberadamente para afectar a la calificación global de una aplicación. El feedback puede estar dirigido a inflar o desinflar la calificación. Algunas personas tratan de inflar las calificaciones por diversas razones, por ejemplo para tratar de conseguir que su aplicación sea promocionada por terceros. Otros pueden dirigirse a las aplicaciones de sus competidores para rebajar sus calificaciones y afectar así negativamente a su atractivo, reducir el número de descargas, etc.

Como ocurre con el correo basura, algunos comentarios de feedback deshonesto pueden ser fáciles de detectar para o bien reportarlos o quitarlos de nuestros análisis, por ejemplo si hay un montón de opiniones idénticas con el mismo texto, etc. Otros pueden estar muy mal escritos. Algunas tiendas de aplicaciones están trabajando intensamente para reducir este tipo de feedback. Análogamente Amazon ha eliminado un montón de comentarios falsos y está demandando a varias personas que los escribieron[5].

4 doi.ieeecomputersociety.org/10.1109/MS.2014.50

5 http://www.wired.co.uk/news/archive/2015-10/19/amazon-fake-reviews-legal-action-fiverr

Redes Sociales

Las redes sociales incluyen servicios como Facebook y Twitter y más recientemente sitios de vídeo sociales como YouTube, donde la gente comparte en línea sus pensamientos, impresiones y sentimientos con varios grupos sociales como amigos, colegas y conocidos. Pueden compartir cosas públicamente, de tal manera que cualquiera puede ver lo que se haya compartido. Algunos de estos comentarios se refieren a aplicaciones móviles. Un buen ejemplo de esto es la propia aplicación para iOS de Facebook, que gastaba la batería del móvil rápidamente. Uno de los responsables de ingeniería de Facebook, Ari Grant, explica las causas y las correcciones en un artículo en línea[6]. Curiosamente, algunos de los comentarios posteriores indican que el problema puede haber vuelto varias versiones después.

Interpretación y Deducción

Como sabemos por experiencia personal, las palabras que escriben las personas no son necesariamente lo que quieren decir o exactamente lo que quieren expresar. Puede que no sepan escribir adecuadamente o que utilicen una gramática pobre. Además, en la interpretación de lo escrito hay matices. Por ejemplo, en los mensajes de texto utilizar un punto para finalizar la frase puede considerarse no sincero[7].

El feedback puede estar en varios idiomas. Las tiendas de aplicaciones pueden utilizar traducción automática para ayudarnos a leer e interpretar, pero de todos modos nuestra comprensión será incompleta e imperfecta. Encontrar a hablan-

6 facebook.com/arig/posts/10105815276466163

7 lifehacker.com/ending-text-messages-with-periods-can-make-them-seem-in-1747411231

tes nativos de otros idiomas puede ayudarnos a trabajar más eficazmente con los usuarios que proporcionan información en lenguas que no son la nuestra. Las analíticas de emociones y sentimientos amplían el feedback en muchas dimensiones, los veremos brevemente.

Minería de Datos

Es posible que tengamos que lidiar con una gran cantidad de texto continuamente. Por ejemplo, algunas aplicaciones populares reciben varios miles de comentarios cada día, y eso es caro de procesar sin utilizar software. La minería de datos puede ser útil para procesar grandes cantidades de datos, y nos ayuda a identificar tendencias y descubrir nuevas ideas a partir de los comentarios que recibimos. La minería de datos es un tema de investigación vivo, e incluso existe un proyecto de investigación académica dedicado a ello llamado UCLappA[8].

También se puede encontrar en línea una introducción gratis a la minería de datos escrita para programadores[9].

Algunos servicios en línea tales como AppAnalytics de Fido Labs[10] tratan de procesar el feedback de las tiendas de aplicaciones "automágicamente", aunque en el momento de escribir esto solo funciona para aplicaciones de iOS.

Analíticas de Emociones

Las emociones pueden ser de gran importancia para algunas aplicaciones, y las emociones de los usuarios pueden afectar a su percepción de una aplicación y al feedback que de ella proporcionen. Existen aplicaciones que aseguran medir las

8 www0.cs.ucl.ac.uk/staff/F.Sarro/projects/UCLappA/home.html

9 /guidetodatamining.com

10 apps.fidolabs.com

emociones del usuario utilizando fuentes de datos visuales[11] y auditivos[12]. Se trata de los primeros indicadores de lo que en un futuro próximo podría ser no solo posible sino además muy útil.

Analíticas de Sentimientos

Las analíticas de sentimientos procesan lo que los usuarios comunican para determinar sus sensaciones, sus sentimientos y lo que desean comunicar a los demás. Estos otros podemos ser nosotros, nuestra organización, sus amigos o cualquier otra persona que descubra lo que los usuarios han comunicado.

Diseñar para el Feedback de la App Móvil

Los desarrolladores más astutos de aplicaciones diseñan sus apps y sus sistemas para fomentar que los usuarios dejen feedback en el momento en que están más predispuestos a dejar buenos comentarios y para redirigir este feedback cuando es probable que sea negativo o crítico. Los usuarios pueden estar de buen humor después de completar con éxito una acción, por ejemplo superar un nivel en un juego o realizar una compra en un sitio de comercio electrónico. Una petición discreta para que dejen un comentario en ese momento puede ser bien recibida por algunos usuarios, y otros pueden simplemente seguir con cualquier otra cosa que quieran hacer. Así mismo, los diseñadores pueden incluir algunas sugerencias de feedback con la esperanza de animar a los usuarios a utilizarlas.

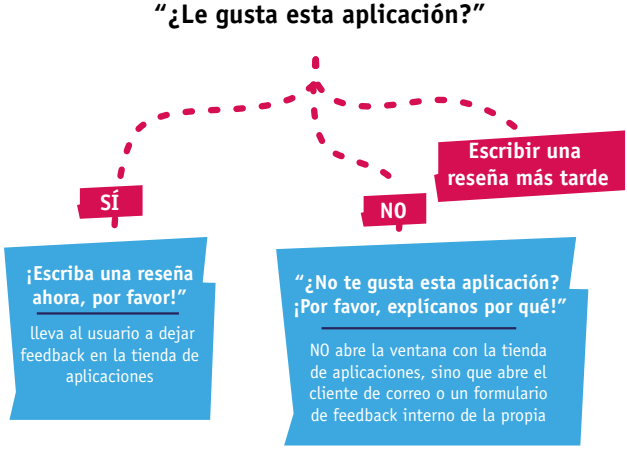
Pedir a los usuarios en tu aplicación que envíen feedback es una forma sencilla de empezar. Sin embargo, al hacerlo intenta no obstaculizar o frustrar a los usuarios, por ejemplo

11 affectiva.com/solutions/mobile

12 beyondverbal.com

no les interrumpas en mitad de un proceso o un nivel de juego, etc. (o al menos no lo hagas si esperas obtener un feedback positivo).

Al implementar un sistema de feedback en tu aplicación, puede interesarte desviar el que sea negativo para que no llegue a la tienda de aplicaciones, en donde sería público y afectaría negativamente a la calificación de la aplicación en la tienda. Un diálogo que ofrezca las siguientes opciones sería una manera de lograr esto:



Algunos equipos de proyecto pueden decidir incorporar servicios de feedback comerciales, tales como HelpShift[13], para ayudarles a gestionar de forma proactiva el feedback de los usuarios de su aplicación.

13 helpshift.com

Respuesta al Feedback

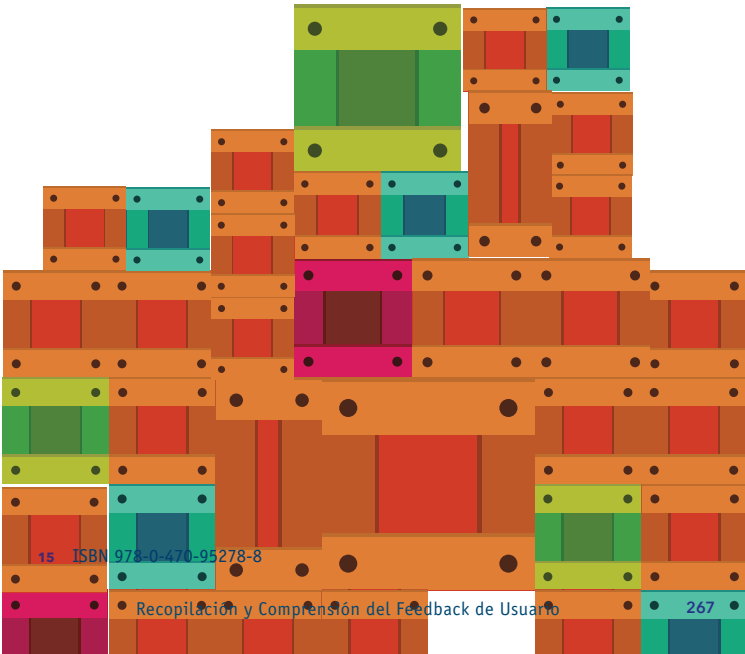
Para muchas organizaciones, el hecho de ser capaz de responder con precisión y rapidez les permite mejorar no solo la percepción del usuario, sino también las percepciones de muchas más personas, ya sea directa o indirectamente (por ejemplo a través del propio usuario hablándoles a otros de las cosas buenas que hemos hecho). Sin embargo, según appbot[14] en Google Play el 97% de opiniones quedan sin respuesta. ¿Quizás puedas superar a muchos de tus competidores respondiendo a todos los comentarios que los usuarios te hagan?

La mayoría de las tiendas de aplicaciones ofrecen la oportunidad de responder a los comentarios de tus aplicaciones. Al responder, sobre todo en el caso de los comentarios negativos, los proveedores de aplicaciones pueden ayudar a sus usuarios y aumentar así su grado de vinculación con la aplicación. Si ayudas a los usuarios a resolver un problema que han encontrado, los usuarios pueden (y con un poco de fortuna así lo harán) cambiar su mala calificación y darle a tu aplicación más estrellas que en su puntuación inicial. **Atención**: No respondas con respuestas de tipo plantilla, de lo contrario puedes dar a los usuarios la impresión de que no estás tratando su comentario con el suficiente respeto (por ejemplo, "Gracias por tu comentario, lo tendremos en cuenta"). En su lugar, haz que el usuario sienta que es especial y su opinión es valiosa incorporando los datos que el usuario haya proporcionado (incluyendo su nombre si está disponible).

14 blog.appbot.co/97-of-google-play-app-reviews-go-unanswered/

Un libro llamado The Art of the App Store[15] dedica un capítulo al feedback. Sus puntos clave son los siguientes:

- Categorizar el feedback, por ejemplo identificando las críticas constructivas. Los comentarios imprecisos pueden ser descartados en esta etapa.
- Convertir el feedback en tareas a realizar, incluyendo correcciones sugeridas y asignación de prioridades.
- Actualizar la aplicación con varias correcciones y mejoras.
- Y, por último, redactar las notas de texto de la próxima versión para que los usuarios puedan leer fácilmente las buenas noticias sobre las mejoras y así instalar las actualizaciones sin demora.



15 ISBN 978-0-470-95278-8





Monetización

Por fin has terminado tu aplicación o sitio web para móviles y la has pulido como resultado del feedback de las pruebas beta. Suponiendo que no estás desarrollando por simple distracción o para promocionar una marca o similar, ha llegado el momento de hacer algo de dinero. Pero, ¿cómo se hace eso y cuáles son tus opciones?

En general, tienes las siguientes vías de monetización:

- **Pago por descarga:** vende tu aplicación por descarga
- **Pago en aplicación:** añade opciones de pago dentro de tu aplicación
- **Publicidad móvil:** gana dinero por publicidad
- **Patrocinios:** recibe un pago cada vez que uno de tus usuarios se da de alta en un servicio de tu patrocinador
- **Participación en ingresos:** obtén ingresos por servicios del operador que se originen en tu aplicación
- **Ventas indirectas:** afiliados, venta de datos y bienes físicos, entre otros
- **Mercado de componentes:** vende componentes o una versión marca blanca de tu aplicación a otros desarrolladores
- **Suscripciones de plataforma de aplicaciones:** crea pequeñas aplicaciones y alquílaselas a empresas

Al planificar tu desarrollo, la determinación del modelo de negocio para la monetización debería ser uno de los elementos clave de su diseño inicial, ya que podría afectar al comportamiento técnico y funcional de la aplicación. El artículo "Five strategies to monetize your mobile app"[1] es una excelente referencia sobre

1 medium.com/@signored/dont-fall-below-the-app-poverty-line-9b800a214e4a

cómo plantear los aspectos financieros para evitar incomodar a los usuarios o perder los ingresos que esperabas conseguir.

Pago por Descarga

Usando el pago por descarga (PPD, en inglés Pay Per Download) tu aplicación se vende una vez a cada usuario al descargarla e instalarla en su teléfono. El pago puede ser gestionado por una tienda de aplicaciones, operador móvil o mediante otro mecanismo configurado por ti mismo. El que una vez fue el método de monetización más popular y rentable hoy es utilizado por una minoría de los desarrolladores. Gartner predice que para el año 2017 casi el 95% de las descargas serán de aplicaciones gratis, frente a aproximadamente el 90% en la actualidad[2].

Cuando tu aplicación se distribuye en una tienda de aplicaciones, la tienda se encargará del mecanismo de pago por ti. A cambio, la tienda cobra una comisión por todas las ventas (habitualmente el 30%). En la mayoría de los casos, las tiendas ofrecen una matriz de precios fijos por país y moneda (0,99 dólares, 0,70 euros, etc.) para elegir a la hora de fijar el precio de tu aplicación.

El pago por la descarga de aplicaciones se gestiona habitualmente de una de las siguientes maneras: o bien se factura a través del operador de telefonía o se efectúa el cobro mediante tarjeta de crédito.

La facturación vía operador permite a tus clientes pagar por tu aplicación con solo confirmar que la venta se cargará a su factura de teléfono móvil o mediante el envío de un SMS premium. En algunos casos, la facturación vía operador se gestiona mediante una tienda de aplicaciones (por ejemplo Google Play,

2 www.businessweek.com/articles/2013-09-19/the-profitable-future-of-free-mobile-apps

que soporta la facturación vía operador para un buen número de operadores por todo el mundo). En otros casos, puede ser implementada directamente por el desarrollador.

Los operadores reciben una parte de los beneficios del precio de venta (normalmente del 30% al 65%, pero algunos pueden llegar a quedarse con el 95%) y si utilizas un agregador también recibirá una parte. Es habitual que la seguridad (cómo evitar que copien tu aplicación) y la manejabilidad sean temas problemáticos en el modelo PPD, pero en ciertos escenarios puede ser la única manera que tendrás de monetizar tu aplicación. La facturación a través de operadores puede resultar una tarea difícil de gestionar por tu cuenta, sobre todo si quieres vender en varios países, ya que es necesario firmar contratos con un operador en cada país. Por razones desconocidas, algunos operadores, como Vodafone, están eliminado la opción de facturar al operador vía Android Play en algunos mercados clave, como el Reino Unido y Alemania. Posiblemente porque hay mejores alternativas, como por ejemplo los pagos bancarios vía móvil.

Vale la pena señalar que la mayoría de los propietarios de tiendas de aplicaciones están llevando a cabo acuerdos de facturación vía operador. La razón principal de esto es que normalmente, cuando los usuarios tienen la opción de comprar con tarjeta de crédito o con métodos de facturación vía operador, muestran una preferencia significativa hacia esta última opción (Nokia dice que sus estudios indican un incremento por diez de los beneficios respecto a los pagos con tarjeta de crédito).

La facturación mediante tarjeta de crédito es utilizada por Apple, Google (en algunos casos), Amazon y otras tiendas. Apple lleva años exigiendo a los usuarios de iPhone que den los datos de una tarjeta de crédito cuando se registran, y Google también ha comenzado a utilizar este método para usuarios de Android. Obtener estos datos antes de que se realice compra

alguna es, según los analistas, un factor diferenciador clave para obtener mayores beneficios mensuales por aplicación.

La última opción de pago consiste en crear tu propio sitio web y poner en práctica un mecanismo de pago a través de él, como PayPal móvil, la marcación a números de teléfono de tarificación especial[3] u otros.

El uso de PPD se puede implementar habitualmente sin un diseño o requisitos de codificación especiales para tu aplicación. Para comenzar, recomendamos utilizar las opciones de facturación de las tienda de aplicaciones, ya que implican un coste mínimo de instalación y menores gastos generales de administración.

Para cada forma de pago es importante determinar la elasticidad en el precio de la demanda o PED[4]. Incrementar el precio no significa necesariamente incrementar los beneficios (y viceversa), ya que tu precio debe encajar con las expectativas de tu base de usuarios.

Pago en Aplicación

El pago dentro de la aplicación (o IAP, en inglés In-App Payment) es una forma de cobrar por acciones o contenidos específicos dentro de tu aplicación. Un uso muy básico podría permitir la compra de tu aplicación (sin suscripciones ni cargos recurrentes) después de un período de prueba, cosa que puede generar más ventas que el sistema PPD si crees que las características de tu aplicación justifican un precio más alto. Alternativamente, puedes ofrecer las características básicas de tu aplicación de forma gratuita, pero cobrar por contenido premium (vídeos, créditos virtuales, información premium,

3 daopay.com

4 en.wikipedia.org/wiki/Price_elasticity_of_demand

características adicionales, eliminación de los anuncios y similares). La mayoría de las tiendas de aplicaciones ofrecen una opción de compra en aplicación o puedes implementar tu propio mecanismo de pago. Si quieres algo más que el pago en una única vez de una "licencia completa", tienes que pensar cuidadosamente acerca de cómo, cuándo y por qué estarán dispuestos a pagar tus usuarios y diseñar tu aplicación en consecuencia. Los pagos en aplicación recurrentes, también llamados subscripciones, también son posibles en la mayoría de las plataformas. Este tipo de pagos son adecuados cuando tu aplicación ofrece contenido que se actualiza de manera regular, como es el caso de los periódicos online o las revistas digitales.

Las compras dentro de las aplicaciones se han convertido en el modelo más próspero de monetización en muchos mercados, particularmente entre los juegos "freemium", que se distribuyen gratuitamente para enganchar a los usuarios antes de convertirlos en clientes. La IAP es especialmente popular en los videojuegos (por ejemplo para comprar energía extra, más niveles, créditos virtuales, etc.) y puede ayudar a conseguir una gran base de usuarios, ya que se ofrece la aplicación gratuitamente.

Distimo informó en el 2014 que las compras in-app suponen el 79% de los beneficios en iOS[5].

Si tu objetivo son países concretos, sé consciente de las posibles diferencias en su comportamiento. Por ejemplo en China la compra inicial supone el 99% del beneficio generado, mientras que la IAP es muy baja; en cambio, en los Estados Unidos ocurre justo lo contrario.

También debería ser obvio que necesitarás diseñar y desar-

5 "2014 How the Most Successful Apps Monetize Globally" disponible en www.distimo.com/publications

rollar tu aplicación de manera que incorpore el método de pago en aplicación. Si la aplicación se ejecuta en varias plataformas, puede ser necesario implementar un mecanismo diferente para cada una de ellas (y, potencialmente, para cada tienda de aplicaciones).

Al igual que ocurre con PPD, te recomendamos que comiences con el pago desde la aplicación ofrecido directamente por el operador o bien con el mecanismo de compra en aplicación ofrecido por una tienda de aplicaciones, especialmente porque algunos de estos servicios pueden incrementar la facturación vía operador (tales como Google Play) o utilizar la información de la tarjeta de crédito preexistente (como Apple o Amazon). Desde el punto de vista de los usuarios, esta es la manera más fácil y más conveniente de pago (uno o dos clics, sin necesidad de introducir números de tarjetas de crédito, nombres de usuario u otras credenciales), por lo que los desarrolladores pueden esperar los mayores grados de aceptación por parte del usuario y las mayores tasas de conversión.

Publicidad Móvil

Como ocurre con los sitios web, puedes decidir ganar dinero mostrando anuncios. De acuerdo con la encuesta de VisionMobile entre 13.000 desarrolladores[6], casi la mitad de los desarrolladores de aplicaciones móviles dependen todavía de este modelo de ingresos, aunque parece probado que solamente es rentable para una pequeña minoría: el 83% de la los participantes en el estudio que confían en los anuncios ganan menos de 10.000 dólares mensuales.

Hay una serie de entidades que ofrecen herramientas para mostrar anuncios para móviles, siendo esta la forma más fácil

6 vmob.me/DE3Q15

de ganar dinero en aplicaciones web móviles. *Admob.com*, *Buzzcity.com* y *inmobi.com* (para videojuegos) son algunas de ellas. Sin embargo, debido a la amplia gama de países, dispositivos y características, en la actualidad hay más de 70 grandes redes de publicidad móvil. Cada red ofrece enfoques ligeramente diferentes y encontrar la que mejor monetice la audiencia de tu aplicación puede no ser tarea simple. No hay una regla de oro, es posible que tengas que experimentar con algunas para encontrar la que mejor te funcione. Sin embargo, para un arranque rápido puedes considerar el uso de un agregador de publicidad móvil como por ejemplo Madgic[7], smaato[8] o inneractive[9], ya que tienden a dar mejores resultados al combinar y optimizar los anuncios de más de 50 redes móviles de publicidad. La mayoría de los agregadores pueden operar como mercados de publicidad ofreciendo Real Time Bidding (RTB), como si fuera una subasta en vivo donde el precio de cada anuncio se determina dinámicamente. La mayoría de las redes de publicidad tienen una comisión del 30% al 50% en los ingresos por publicidad y los agregadores de un 15% a un 20% adicional, pero incluso con esas cifras los agregadores son más rentables que intentar integrar diversas redes de publicidad por ti mismo.

Si tu aplicación está dando buenos resultados y tiene un gran volumen en un determinado país, podrías considerar vender publicidad directamente a agencias de publicidad o marcas (publicidad premium) o contratar a una agencia de medios para hacer eso por ti.

También en este caso, muchos de los fabricantes de dispositivos ofrecen servicios de publicidad móvil como parte

7 madgic.com

8 smaato.net

9 inner-active.com

de las funciones de su tienda de aplicaciones, e igualmente vale la pena explorar estos mecanismos. En algunos casos puede que tengas que utilizar los servicios de publicidad del proveedor para poder incluir tu aplicación en su tienda.

La publicidad dentro de la aplicación requiere que diseñes y programes tu aplicación con cuidado. No solo debes considerar con cuidado la ubicación de los anuncios en tu aplicación, sino también las variaciones y el mecanismo de exclusión (opt-out). Si los anuncios se vuelven demasiado intrusivos puede que los usuarios abandonen tu aplicación, mientras que hacer la publicidad demasiado sutil significará pocos o nulos ingresos. Relativamente nueva, comparada con la publicidad tradicional de banners, es la publicidad intersticial: este término se usa generalmente para describir un anuncio que ocupa toda la pantalla y que suele ir acompañado de un botón "skip screen". Otros nuevos formatos incluyen los anuncios a los que se puede jugar y los anuncios con premios, especialmente en los videojuegos. Puedes necesitar experimentar un poco para encontrar la manera y posición correcta en la que colocar los anuncios.

Patrocinios

La startup alemana Apponsor[10] ofrece una nueva manera de ganar dinero sin tener que mostrar publicidad o cobrar por descargar la aplicación: el usuario obtiene tu aplicación de manera gratuita y se le pide darse de alta en un boletín electrónico de tu patrocinador. El patrocinador te paga por cada nuevo registro en ese boletín. Conviene no confundir este sistema con las empresas de tipo App Sponsors, que son empresas que pagan el coste de desarrollo de la aplicación pidiendo a cambio

10 apponsor.com

una participación en los beneficios. Para más información entra en Apps Funder[11].

Ventas Indirectas

Otra opción es utilizar tu aplicación para dirigir las ventas donde desees.

Con este sistema, normalmente ofreces tu aplicación o sitio web de manera gratuita y empleas mecanismos tales como:

1. **Programas de afiliación**: haz promoción de aplicaciones de pago propias o de terceros en el interior de una gratuita. MobPartner.com es un proveedor de servicios que ofrece este tipo de monetización.
2. **Venta de datos**: haz seguimiento del comportamiento de tus usuarios y vende los datos a partes interesadas. Ten en cuenta que, por razones de privacidad, no debes revelar ninguna información personal, así que debes asegurarte de que todos los datos se proporcionan en informes anónimos y consolidados.
3. **Mundo real versus virtual**: utiliza tu aplicación como herramienta de marketing para vender bienes en el mundo real. Ejemplos típicos son aplicaciones de vehículos, de revistas y de grandes marcas como McDonald's y Starbucks. También utilizan este modelo de negocio aplicaciones de cupones como Groupon. {Solo el 10% de los participantes en el informe de DevEconomics del tercer trimestre del 2015 están confiando en este modelo de negocio, aunque los desarrolladores de aplicaciones de comercio tienen más del triple de probabilidades de ganar más de

11 appsfunder.com

100.000 dólares mensuales que aquellos monetizando con *publicidad. vmob.me/DE3Q15*

No hay nada que te impida combinar esta opción con cualquiera de las otras de monetización si lo deseas, pero ten la precaución de no dar la impresión de realizar promociones extremadamente intrusivas.

Mercado de Componentes

Un mercado de componentes (CMP, en inglés Component Marketplace) proporciona otra oportunidad para que los desarrolladores moneticen sus productos mediante la venta a otros desarrolladores de componentes de software o aplicaciones de marca blanca. Un componente es una pieza de construcción de software que ofrece una funcionalidad definida para ser utilizado por el software de nivel superior.

La típica pregunta que surge en este punto es cómo los CMPs conviven con el código abierto. Como usuario, el código abierto es a menudo gratuito, el código fuente debe ser proporcionado y los usuarios tienen el derecho de modificarlo y distribuir su obra derivada.

Algunos proveedores de componentes requieren el pago de una licencia. Pueden proporcionar el código fuente completo que permita al desarrollador depurar código en el nivel más bajo. Algunos CMPs soportan todos los modelos: componentes de pago o componentes libres, en ambos casos con o sin el código fuente.

Si eres un desarrollador en busca de un componente, los CMPs ofrecen dos ventajas principales. En primer lugar, no tienes que liberar tu código fuente solo porque utilices componentes de software. Todo el código abierto viene con una licencia. Algunas licencias como la Apache te permiten comercializar el producto que hagas con ellas; otras, como AGPL y OSL, requieren que liberes

tu código fuente que integra el suyo. Es posible que no quieras esto. En segundo lugar, los CMPs proporcionan una forma fácil de encontrar y descargar componentes. En cambio puedes pasar días mirando los repositorios de código abierto para encontrar el componente adecuado.

Los mercados de componentes han existido desde hace décadas. El mercado más importante es el de componentes para .NET y Visual Basic en la comunidad Windows. Mercados como componentOne y proveedores como Infragistics son bien conocidos en sus ámbitos. En cambio, la idea de los mercados de componentes dentro del entorno móvil es bastante nueva. ChupaMobile[12] es un actor relevante en este ámbito.

Suscripciones de Plataforma de Aplicaciones

La segunda oleada tras la adopción de aplicaciones móviles por parte de los consumidores vino de las pequeñas y medianas empresas. Según Gartner[13] la capacidad de desarrollo necesario para satisfacer la demanda de aplicaciones empresariales será crítica en 2017.

Las empresas más pequeñas, incluyendo tu panadería de la esquina, están interesadas en tener su propia aplicación, pero no tienen el presupuesto para justificar su desarrollo. Las plataformas de aplicaciones son una excelente opción para este mercado. El desarrollador diseña una aplicación con algunas opciones por defecto, añade el contenido y vende una suscripción a la empresa. Todo el contenido está alojado en línea, solo el chasis de la aplicación se descarga desde las tiendas de aplicaciones. Por lo que la empresa tiene que renovar su suscripción cada mes o cada año para

12 www.chupamobile.com

13 www.gartner.com/newsroom/id/3076817

mantener su aplicación funcionando. Dominar las herramientas de desarrollo de aplicaciones de la plataforma y vender un par de nuevas suscripciones mensuales aseguran al desarrollador un ingreso recurrente.

Eligiendo tu Modelo de Monetización

Así que, con todas estas opciones, ¿cuál debería ser tu estrategia? Depende de tus objetivos, veamos algunos:

- ¿Estás seguro de que los usuarios desearán comprar tu aplicación inmediatamente? Entonces véndela como PPD a 0,99 dólares pero ten en cuenta que, aunque puede que ingreses varios miles de dólares diarios, fácilmente pueden no ser más que algunos cientos semanales si tu apreciación de la aplicación es poco apropiada o la competencia es feroz. La Application Developer's Alliance recomienda el método de monetización PPD para aplicaciones de producción costosa, con barreras de entrada o aplicaciones de elevado volumen[14]. Esto incluye juegos y aplicaciones de entretenimiento, productividad, navegación y noticias.
- ¿Estás apuntando a una gran base de usuarios? Considera distribuir la aplicación de forma gratuita con compras dentro de la aplicación o con publicidad móvil (incluso podrías ofrecer una versión premium sin publicidad).
- ¿Estás ofreciendo capacidades premium a precio premium? Considera una versión limitada en tiempo o capacidades y usa las compras dentro de la aplicación para permitir la compra de una versión completa, ya sea permanentemente o por un período de tiempo.
- ¿Estás desarrollando un juego? Considera ofrecerlo de

14 www.appdevelopersalliance.org/app-monetization

manera gratuita con publicidad, o bien una versión básica que incluya compras internas para desbloquear nuevas funciones, más niveles, distintos vehículos o cualquier equipamiento del juego.

- ¿Es tu aplicación móvil una extensión de una tienda web o física? Ofrece la aplicación gratis y gana ingresos de los productos y servicios que comercializas en el mundo real.
- ¿Tiene tu aplicación contenido que se actualiza frecuentemente, como una revista digital? Ofrece pagos recurrentes y asegúrate de que lo usuarios vuelven.
- ¿Ofreces bienes físicos, como la aplicación de una tienda? Ofrece la aplicación gratuitamente y obtén comisiones de las compras de los clientes.

Estrategias para las Tiendas de Aplicaciones

La otra cara de la generación de ingresos es el marketing y la promoción. La necesidad puede ser obvia si vendes tu aplicación a través de tu propio sitio web, pero resulta igualmente importante cuando se usa la tienda de aplicaciones de un vendedor. Esas tiendas son a la vez la bendición y la maldición de los desarrolladores móviles. En el lado positivo, dan a los desarrolladores un largo alcance y una exposición potencial de ventas que de otra manera sería muy difícil de lograr. En el lado negativo, las más populares contienen actualmente cientos de miles de aplicaciones, disminuyendo así el potencial para destacar entre la multitud y tener éxito, llevando a muchos a comparar la posibilidad de tener éxito en una tienda de aplicaciones a la probabilidad de ganar la lotería.

Dicho esto, aquí están algunos consejos y trucos para ayudarte a incrementar tus probabilidades.

Estrategias para Conseguir un Ranking Alto

Lo más importante que debes comprender acerca de las tiendas de aplicaciones es que son canales de distribución y no máquinas de marketing. Esto significa que aunque son una gran manera de conseguir introducir tu aplicación en los dispositivos de los usuarios, no van a promocionar tu aplicación por ti (a menos que compres posicionamiento premium, ya sea a través de banners o posiciones de lista). No puedes confiar en las tiendas de aplicaciones para incrementar tus descargas, a menos que aparezcas en una lista de aplicaciones de más éxito. Pero no debes jugar a la lotería con tus aplicaciones: ten una estrategia y planifica cómo comercializar tu aplicación.

Hemos preguntado a muchos desarrolladores acerca de las tácticas que les han proporcionado la mayor atención y un mejor ranking en las tiendas de aplicaciones.

Nos dieron muchas respuestas y salió a la luz un tema común: no hay ninguna fórmula mágica, ¡tienes que atacar por todos los frentes! Sin embargo, te ayudará tratar de mantener en cuenta lo siguiente:

- Necesitas una aplicación rompedora: debería ser entretenida, fácil de usar y estar libre de errores. Asegúrate de ponerla en manos de usuarios de prueba antes que en las tiendas.
- Perfecciona tus iconos e imágenes para la tienda de aplicaciones, redacta una buena descripción de tu aplicación y elige cuidadosamente las palabras clave y categorías. Si estás inseguro o insatisfecho con los resultados, experimenta.
- Obtener reseñas de blogueros y revistas es una de las mejores maneras de recibir atención. Como contraprestación, algunos te pedirán dinero, otros exclusividad y otros acceso anticipado a tus aplicaciones.
- Consigue críticas positivas tan pronto como sea posible.

Llama a tus amigos y pide a tus usuarios una reseña de manera regular.

- Si vas a llevar a cabo cualquier tipo de publicidad, hazlo en avalancha durante un par de días. Esto es mucho más efectivo que gastar la misma cantidad de dinero durante dos semanas, ya que te ayudará a alcanzar un elevado pico de audiencia, más que una subida lenta y gradual.
- No confíes en el tráfico generado por personas que exploran la tienda de aplicaciones, asegúrate de conducir tráfico a tu aplicación a través de tu sitio web, SEO y social media.

Múltiples Tiendas versus Tienda Única

Con más de 120 tiendas de aplicaciones disponible para los desarrolladores, evidentemente hay muchas opciones para distribuir una aplicación. Pero los 20 minutos necesarios en promedio para enviar una aplicación a una tienda significan que podrías pasar mucho tiempo publicando aplicaciones en tiendas desiertas que consiguen pocas descargas. Esta es la razón por la que la mayoría de los desarrolladores se adhieren a solo una o dos tiendas, perdiendo así una oportunidad potencialmente enorme, pero consiguiendo mucho más tiempo para cosas importantes, ¡como la programación! Así que, ¿debes centrarte en múltiples tiendas o no?

Múltiples tiendas	Tienda única
Las tiendas de aplicaciones de las principales plataformas pueden tener serias limitaciones, como los mecanismos de pago, la penetración en determinados países y las directrices de contenido.	Más del 90% de los usuarios solo utilizan una única tienda de aplicaciones, que suele ser la que viene por defecto incluida con el teléfono.

Múltiples tiendas	Tienda única
Las tiendas más pequeñas te dan más opciones de visibilidad (por ejemplo siendo la aplicación destacada).	Tu propio sitio web te puede aportar más tráfico que las tiendas de aplicaciones (especialmente si tienes una marca conocida).
Las tiendas más pequeñas tienen un entorno social más amistoso que las grandes.	Muchas tiendas pequeñas arrastran datos desde las grandes, así que puede que tu aplicación esté ya en ellas.
Las tiendas de los operadores tienen directrices para contenidos notoriamente estrictas y puede ser difícil entrar, particularmente para algunos tipos de aplicaciones.	Para contenidos sin nicho, las tiendas de operadores o plataformas pueden ofrecer suficiente exposición para no justificar el esfuerzo extra de una estrategia de múltiples tiendas.
Las tiendas más pequeñas pueden ofrecer un amplio rango de opciones de pago y modelos de negocio o estar disponibles en muchos países.	Algunas tiendas de operadores tienen procesos de facturación más fáciles, como la facturación directa a la cuenta de teléfono del usuario, conduciendo a un incremento en la ratio de conversión.
Algunos desarrolladores reportan que el 50% de sus beneficios en Android vienen de fuera del Android Market.	Los desarrolladores para iOS solo necesitan una tienda.

Las tiendas de aplicaciones deberían darte acceso global a los usuarios, pero con el paso del tiempo te será interesante adaptar tu estrategia respecto a ellas para que encaje con tu base de usuarios objetivo y utilizar solamente las que se

adapten más a ese público. Esto podría implicar usar tiendas particulares de operadores, tiendas populares en un determinado país o sencillamente mantenerte fiel a las tiendas de las plataformas. Existen algunas tiendas de terceros con un gran volumen de audiencia, como por ejemplo la tienda de Amazon para Android, que ofrecen a los desarrolladores un buen número de maneras de monetizar sus aplicaciones, por ejemplo PPD y pagos en aplicación para muchos países. Además, en algunos países existen tiendas de aplicaciones locales muy populares, por ejemplo la alemana AndroidPit[15] o algunas dedicadas a Android en China.

¿Cuánto Puedes Ganar?

Una de las preguntas más habituales entre desarrolladores es acerca de cuánto dinero se puede ganar con una aplicación móvil. Está claro que algunas aplicaciones han hecho millonarios a sus desarrolladores, mientras que otros no podrán renunciar a su puesto de trabajo en el corto plazo. La mayoría de los desarrolladores de aplicaciones no están generando suficientes ingresos ni siquiera para cubrir los costes del desarrollo y los desarrolladores dedicados a una única plataforma confirman que los ingresos generados por su aplicación de más éxito no resultan suficientes para sustentar un negocio independiente. Según VisionMobile, más del 50% de ellos se encuentran por debajo del "límite de pobreza por aplicación" fijado en 500 dólares mensuales para cada una[16].

Los juegos móviles son los que parecen ofrecer más opciones

15 androidpit.de

16 www.vmob.me/DE3Q15

para hacer dinero, según Pulse[17]. Y pese a que los teléfonos y tablets Android superan en número a los dispositivos iOS, los ingresos generados en la App Store de Apple son un 80% más altos que los generados en Google Play[18].

En última instancia, lo que puedes ganar depende de satisfacer una necesidad y de un marketing eficaz. La experiencia sugiere que las aplicaciones que ahorran dinero o tiempo al usuario son más atractivas (descuentos en hoteles, cupones, música gratuita y similares), seguidas por los juegos (basta con ver el éxito de Angry Birds) y las herramientas de negocio (los visores de documentos de oficina, herramientas de sincronización, herramientas de copia de seguridad y otras), pero a menudo el éxito (económico) de una aplicación individual no se puede predecir. El éxito, por lo general, viene con un punto de experimentación y mucha perseverancia.

17 www.linkedin.com/pulse/mobile-gaming-monetization-making-more-money-your-app-jehan-damji

18 http://techcrunch.com/2015/10/15/ios-app-store-revenue-now-80-percent-higher-than-google-play-thanks-to-china/ techcrunch.com/2015/10/15/ios-app-store-revenue-now-80-percent-higher-than-google-play-thanks-to-china]


GAME
OVER

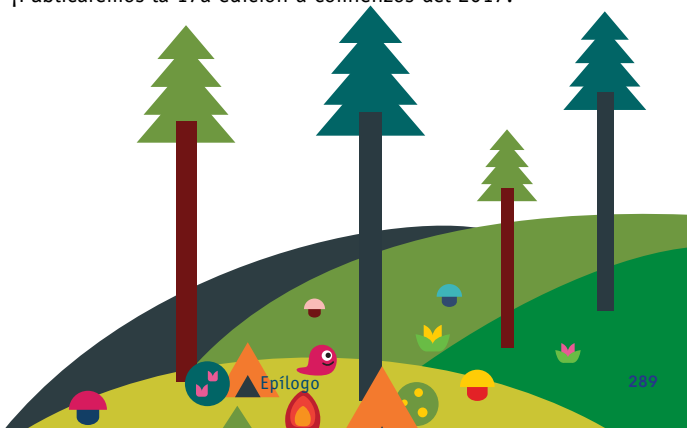
Epílogo

Gracias por leer esta 16a edición de nuestra Guía del Desarrollador Móvil. Esperamos que hayas disfrutado de su lectura y que te hayamos ayudado a clarificar tus opciones. Tal vez ahora estás listo para participar en el desarrollo de una aplicación móvil o hayas descubierto nuevas opciones en el negocio de las aplicaciones. Esperamos que sí. Por favor, involúcrate también con la comunidad y comparte tus experiencias e ideas con nosotros y el resto de compañeros.

Si quieres contribuir a esta guía, patrocinar próximas ediciones o te interesa obtener ediciones anteriores, por favor escríbenos a . Si utilizas Twitter, te invitamos a seguirnos en *twitter.com/mobiledevguide*.

Por supuesto, puedes descargarte esta edición en formato ebook en *amazon.com*, *kobobooks.com* o *apple.com/iBooks*, o alternativamente en formato PDF desde nuestro sitio web *www.enough.de/mdgg*, donde también encontrarás más información sobre Enough Software, la agencia de aplicaciones alemana que coordina este proyecto desde que comenzó en 2009.

¡Publicaremos la 17a edición a comienzos del 2017!



Acerca de este Libro

Este proyecto fue iniciado por Enough Software en 2009 con el objetivo de difundir el conocimiento acerca de las tecnologías móviles y para animar a la gente a entrar en nuestra comunidad o ampliar sus conocimientos previos. Hemos regalado más de 80.000 copias en papel en eventos en todo el mundo. Universidades y escuelas en Alemania, Países Bajos, Reino Unido, España y Sudáfrica utilizan el libro como material didáctico. Las versiones electrónicas (ebook y pdf) se han descargado cientos de miles de veces y el contenido ha sido traducido a varios idiomas. El libro es un proyecto sin ánimo de lucro: los escritores, editores, traductores y diseñadores contribuyen con su trabajo de forma gratuita. Los costes de impresión y distribución se cubren con patrocinadores.

El Editor

Enough Software

Enough Software es una agencia de aplicaciones de Bremen, en el noroeste de Alemania, con más de 10 años de experiencia. Nuestro equipo de diseño de 25 expertos diseñan y desarrollan aplicaciones para cualquier tipo de negocio y para cualquier plataforma. Además siempre estamos encantados de compartir nuestra experiencia, ya sea con este libro, como consultores o como profesores. Valoramos la comunicación abierta y transparente tanto dentro de los equipos de proyectos como trabajando con nuestros clientes. Empresas como la BBC, Nokia, Vodafone, Samsung y CEWE utilizan nuestras herramientas para sus proyectos o trabajan con nosotros como su socio de desarrollo de aplicaciones.

www.enough.de

@enoughsoftware

El Patrocinador de la Versión Impresa

Microsoft

Microsoft es una empresa tecnológica cuya misión es impulsar a cada persona y cada organización en el planeta para lograr sus objetivos. Fundada en 1975, nuestra estrategia es construir las mejores plataformas de cada clase y servicios de productividad para un mundo donde el móvil y la nube son prioritarios. Desarrollamos, patentamos y damos soporte a una amplia gama de productos de software, servicios y dispositivos que ofrecen nuevas oportunidades, una mayor comodidad y un mayor valor a la vida de las personas. Ofrecemos una variedad de servicios, incluyendo servicios basados en la nube, a consumidores y empresas. Diseñamos, fabricamos y vendemos dispositivos que se integran con nuestros servicios basados en la nube, y proporcionamos publicidad en línea relevante a una audiencia global.

www.microsoft.com

Los Autores y Colaboradores

Anna Alfut

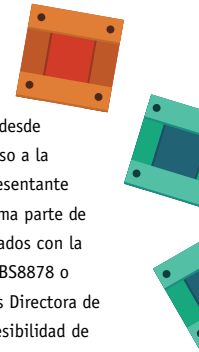
Anna comenzó su vida profesional como diseñadora creativa. Después de descubrir su pasión por el diseño de interfaces, se convirtió en la coautora de una aplicación para iOS y Android y consultora en múltiples proyectos, tanto en el lado agencia como en el de cliente. Actualmente trabaja como diseñadora UX para dispositivos móviles. Aparte de dibujar y pensar a través de interfaces, también hace ilustraciones y disfruta de vivir en Londres.

www.alfutka.net

Davoc Bradley / MiraLife

Davoc ha estado trabajando como ingeniero de software desde 1999, especializándose en arquitectura y diseño de webs y sistemas móviles de alta usabilidad. Actualmente es CTO en MiraLife, proveedores de software web y móvil que tiene por objetivo mejorar las vidas de personas con demencia y otras enfermedades terminales. Davoc es también un músico entusiasta, fan ávido del cricket y le encanta viajar.

@davocbradley



Sally Cain / RNIB

Sally ha trabajado en RNIB en el ámbito de la accesibilidad digital desde hace más de 16 años. Cree apasionadamente en la igualdad de acceso a la tecnología digital para las personas con discapacidad. Sally es representante de su organización en grupos de estándares del W3C, y también forma parte de una serie de grupos en el British Standards Institute (BSI) relacionados con la estandarización de las TIC, lo que incluye el grupo responsable del BS8878 o Código de Prácticas para la Accesibilidad Web. Actualmente, Sally es Directora de Tecnología de Accesibilidad del RNIB, siendo responsable de la accesibilidad de todos los sistemas tanto internos como de cara al cliente y asegurando que el RNIB está cumpliendo con la accesibilidad no solo para los clientes sino también para el personal interno. También ha dirigido la redacción del estándar interno del RNIB para la accesibilidad.

@sallycain

www.rnib.org.uk

Dean Churchill / AT&T

Dean trabaja en el diseño de seguridad, desarrollo y prueba de aplicaciones de AT&T. Desde hace años está centrado en impulsar los requisitos de seguridad en las aplicaciones móviles, tanto para aquellas de consumo como para internas de AT&T. También ha apoyado las líneas emergentes de producto de AT&T en Salud Móvil y Vida Digital. Vive en el área de Seattle y disfruta haciendo esquí alpino y pesca con mosca.

Julian Harty / Commercetest

Julian fue contratado por Google en 2006 como su primer ingeniero de pruebas en Europa responsable de probar las aplicaciones móviles de Google. Ayudó a otros, tanto dentro como fuera de Google, a aprender a hacer lo mismo, y terminó escribiendo el primer libro sobre ese tema. Posteriormente trabajó para eBay, donde su misión era reformar las pruebas a nivel mundial. Actualmente trabaja de forma independiente, escribe aplicaciones móviles y herramientas de automatización de pruebas, y ayuda a otros a mejorar sus aplicaciones móviles. También está escribiendo un nuevo libro sobre testeo y automatización de pruebas para aplicaciones móviles.

@julianharty



Oscar Clark / Unity Technologies

Oscar Clark es un autor, consultor y divulgador de Everyplay (Unity Technologies). Ha sido pionero en el mundo de los juegos sociales online, en móvil y en consolas desde 1998. Oscar ha aportado 'visión' para una de las primeras comunidades de juegos en línea (Wireplay - British Telecom); fue líder mundial en la división de videojuegos en Hutchison Whampoa (3UK), incluyendo la que, posiblemente, fuera la primera compra in-app de la historia; y fue el Home Architect para PlayStation®Home. Actualmente es columnista regular en PocketGamer.Biz y Develop-Online, conferenciante en innumerables conferencias sobre videojuegos, mentor en la aceleradora GameFounders, y profesor invitado en múltiples universidades. Su primer libro, "Games As A Service - How Free To Play Design Can Make Better Games" está disponible online.

@athanateus

www.gamesasaservice.net

Ovidiu Iliescu / Enough Software

Tras desarrollar aplicaciones de sobremesa y web durante varios años, Ovidiu decidió que el software móvil era más de su agrado. Está involucrado en desarrollos Java ME y BlackBerry para Enough Software desde 2009. Se emociona con cualquier cosa relacionada con una programación eficaz, algoritmos y gráficos por ordenador.


@ovvyblabla


www.ovidiuiliescu.com & www.enough.de

Alex Jonsson / Evothings

A Alex le gusta todo lo móvil, tanto aplicaciones como tecnologías web, y especialmente conectar inteligentemente objetos físicos a otros móviles. Tiene un Doctorado en Tecnología de Medios por el Real Instituto Tecnológico de Estocolmo y comparte sus ideas y conocimientos tanto a nivel académico como corporativo. El Dr. Jonsson tiene también una necesidad ecléctica de investigar cómo las aplicaciones y servicios pueden generar nuevos negocios, aportando nuevos valores y maneras de conectar el mundo de manera inteligente. Es el fundador y VP Community de Evothings simplemente porque las cosas son mejores si están conectadas.


@dr_alexj


www.evothings.com

Michael Koch / Enough Software

Michael viene desarrollando software desde 1988, uniéndose al equipo de desarrollo de Enough Software en 2005. Ocupa el cargo de Director de Tecnología. Ha dirigido numerosos proyectos de desarrollo de aplicaciones móviles (principalmente para Java ME, Android, Windows Mobile y BlackBerry), y también es un experto en tecnologías de servidor. Michael es un entusiasta del código abierto involucrado en muchos proyectos libres, como GNU classpath.


@linux_pinguin


www.enough.de

Daniel Kranz / Joule

Daniel es un estratega multicanal con experiencia en consultoría, agencias y tecnología. Anteriormente director técnico de proyectos en una de las principales agencias de publicidad y consultor de soluciones móviles para una consultora especializada en web y multicanal, ahora trabaja como consultor de planificación estratégica global aconsejando a marcas sobre cómo integrar lo móvil como parte de su estrategia digital.

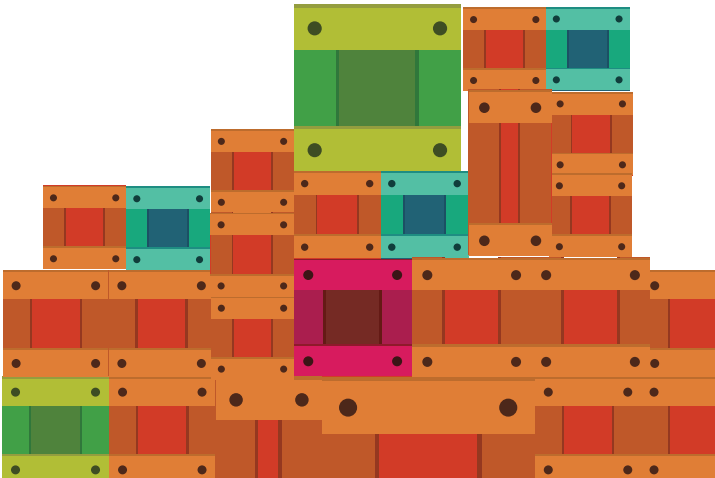
www.jouleww.com

Vikram Kriplaney

Vikram ha sido desarrollador móvil desde que tecnologías como WAP, Symbian y J2ME todavía estaban de moda. Fundó la parte móvil de local.ch en 2007, donde pasó a desarrollar en solitario sitios web para móviles y aplicaciones iOS y Android de gran éxito. Ahora es Arquitecto Móvil y el ingeniero principal de local.ch y search.ch (Swisscom Directories), donde ayudó a construir uno de los equipos de ingeniería móvil más impresionantes de Suiza.

@krips

local.ch iPhonso.com



Carlo Longino / WIP

Carlo ha dedicado más de una década de experiencia la industria móvil, comenzando justo tras el cambio de milenio en la sede de Nokia en Finlandia. Antes de incorporarse a WIP (Wireless Industry Partnership) como director de los servicios de marketing para desarrollo en el 2010, Carlo trabajó como consultor freelance y escritor hasta que completó un MBA. Previamente, fue analista sénior para Floor64, una firma de análisis con sede en Silicon Valley, donde se especializó en las industrias móviles y de telecomunicaciones. También ayudó a lanzar, e impulsó durante cinco años, TheFeature.com, un sitio centrado en el liderazgo de ideas propiedad de Nokia. Carlos también ha publicado en las redes de noticias de The Wall Street Journal, Business 2.0 y Dow Jones, y ha dado conferencias en numerosos eventos, incluyendo el Mobile World Congress, SXSW, MobileBeat y CTIA, entre otros.


@caaarlo


www.wip.org

Albert Mata / Universitat Oberta de Catalunya

Albert es el traductor al castellano de esta edición de la guía. Actualmente se dedica al desarrollo de aplicaciones para iOS y tvOS, aunque en el pasado también ha trabajado con plataformas para el desarrollo multiplataforma como Xamarin. Además ejerce de articulista tratando temas de tecnología orientados a públicos no técnicos.


@almata

Javier Melenchón / Universitat Oberta de Catalunya

Javier Melenchón se dedica a realizar actividades formativas asociadas a la creación multimedia desde el ámbito universitario, inicialmente en la Universitat Ramon Llull y actualmente en la Universitat Oberta de Catalunya. Forma parte de la primera promoción de titulados universitarios en ingeniería multimedia y se doctoró en ese mismo ámbito. Es experto en todo tipo de tratamiento de señales de audio, imagen y vídeo y en cómo formar a los profesionales para que sepan manejar este tipo de información, de crucial importancia en el ámbito móvil actual. Colabora con la revista digital Mosaic.

www.mosaic.uoc.edu & multimedia.uoc.edu

Tim Messerschmidt / PayPal

Tim ha estado desarrollando aplicaciones Android desde 2008. Después de estudiar informática de negocios, se unió a Neofonie Mobile con sede en Berlín como desarrollador de software móvil en 2011, siendo consultor para Samsung Alemania como Developer Advocate para Android y bada desde 2010. En 2012 se trasladó a PayPal como un Developer Evangelist. Es un apasionado de los pagos por móvil, las interfaces de usuario, UX y el desarrollo en Android en general. Además, le encanta hablar en conferencias, escribir artículos y participar en todo tipo de social media.

@seraandroid & @PayPalDev

www.timmesserschmidt.de

Sebastian Meyer / D-LABS

Sebastián tiene más de una década de experiencia con las tecnologías web y móviles. Se unió a D-LABS como Consultor de Innovación y Software después de cursar sus estudios en ingeniería de software en el Hasso Plattner Institute en Potsdam. Está especializado en metodologías centradas en el usuario y la innovación en un contexto empresarial, y trabaja como consultor para startups nacionales e internacionales, PYME y grandes empresas.

www.d-labs.com

Alex Repty

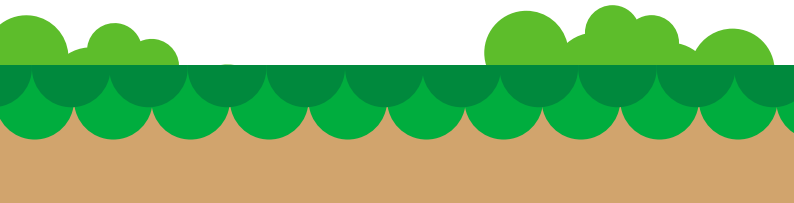
Alex es un ingeniero de software freelance especializado en las plataformas OS X, iOS, watchOS y tvOS. Ha estado desarrollando software para las plataformas de Apple desde que consiguió su primer Mac en 2004. Desde entonces, ha ayudado a crear una amplia variedad de aplicaciones, algunas de las cuales fueron incluso destacadas en la campaña "There's an app for that" de Apple o ganaron un Apple Design Award. Su pasión por el código limpio, las tendencias de ingeniería de software y el diseño de experiencia de usuario hacen que participe como ponente en varias conferencias relacionadas con iOS.

@arepty

André Schmidt / Enough Software

André lleva en el negocio del software desde 2001. Después de iniciar su carrera como programador en una de las empresas líderes en la industria de defensa, se unió a Enough Software en 2007 como desarrollador móvil. En este rol, ha creado una amplia gama de aplicaciones móviles, sobre todo para Android. También es un ponente frecuente en conferencias de desarrolladores y bar camps.

www.enough.de



Michel Shuqair / AppValley

Tras comenzar con aplicaciones WAP en blanco y negro, iMode y juegos SMS en la década de los 90, Michel pasó a liderar la red social Wauwee. Sirviendo a casi un millón de miembros, Michel fue apoyado por un equipo de especialistas en Symbian, iPhone, BlackBerry y Android en su sede en Amsterdam. Wauwee fue adquirida por MobiLuck, ahora parte de la empresa con sede en Paris Madgic.com, una plataforma de monetización móvil.

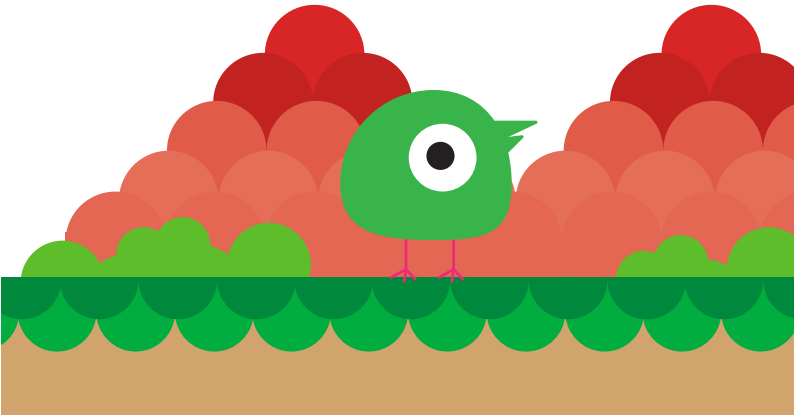
www.appvalley.nl

Marco Tabor / Enough Software

Marco es el responsable de relaciones públicas, ventas y mucho más en Enough Software, donde lleva trabajando desde 2008. El es el coordinador de esta guía, teniendo además la responsabilidad de encontrar patrocinadores y fusionar las aportaciones hechas por la comunidad móvil.

@enoughmarco

www.mobiledevelopersguide.com & www.enough.de



Ian Thain / SAP

Ian es un Mobile Evangelist de SAP, aunque comenzó hace 12 años con Sybase Inc. Se dirige regularmente a público de todo el mundo, proporcionando conocimiento móvil y experiencia corporativa. También escribe artículos, blogs y tuits en movilidad empresarial y es un apasionado de la experiencia en desarrollo móvil en el mundo empresarial.


@ithain


scn.sap.com/blogs/ithain/ & www.sap.com

Marc van 't Veer / Polteq

Marc es un consultor especializado en pruebas de aplicaciones móviles en Polteq y ha trabajado en diferentes roles de pruebas durante más de 9 años. Tiene experiencia en realizar pruebas orientadas a un contexto tecnológico, por ejemplo en telecomunicaciones, SOA, automatización de pruebas y API de pruebas. Actualmente Marc da soporte a las empresas en la mejora de las pruebas de aplicaciones móviles basadas en el enfoque TI4 Mobile.


@marc_vantveer


marcvantveer.niobe.nl & www.polteq.com

Robert Virkus / Enough Software

Robert ha estado trabajando en el espacio móvil desde 1998. Experimentó la fragmentación de Java de primera mano cuando desarrolló y portó un cliente móvil en el Siemens SL42i, el primera teléfono para el mercado de masas con una máquina virtual Java. Después de esta experiencia puso en marcha el proyecto de código abierto J2ME Polish en 2004. J2ME Polish ayuda a los desarrolladores a superar la fragmentación por dispositivo. Es el fundador y CEO de Enough Software, la compañía detrás de J2ME Polish, muchas aplicaciones móviles, y este libro.


@robert_virkus


www.j2mepolish.org & www.enough.de

Chris Ward / Sitepoint

Chris es un desarrollador y escritor trotamundos que trabaja actualmente en varios proyectos con el objetivo de explorar el potencial de la 'cultura libre'. Actualmente es el editor de www.sitepoint.com/mobile y siempre está buscando nuevos escritores.

 *@ChrisChinch*

 *chrischinchilla.com*





Síguenos también en
Twitter *@MobileDevGuide*.
Gracias!

una iniciativa de:



www.enough.de

esponsorizado por:



msdn.microsoft.com

Una visión independiente, no comercial e impulsada por la comunidad sobre tecnologías móviles para desarrolladores y todas las personas que toman decisiones

"Si quieres empezar tu propio negocio en el ámbito móvil, este es el libro que tienes que leer primero." Minh Nguyen on amazon.com

„The title is grand, but the content does live up to the title to an extent. This ebook is a wide-ranging look at almost anything related to mobile app development out there." Azzief Khaliq on hongkiat.com

"¡Una obra espectacular! Te asombrarás por cómo de increíblemente rápido puedes establecer tu presencia en el mercado móvil con los simples pasos explicados en esta guía." Daniel Hudson on webtechman.com

Una lectura obligatoria para cualquiera que esté considerando crear y comercializar apps." Steve on amazon.com

www.mobiledevelopersguide.com